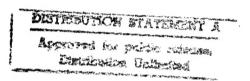
# LOW-FREQUENCY RESONANCE SCATTERING FROM ACOUSTICALLY COMPACT BUBBLE CLOUDS

A Report to the ONR Under the Support of Grant N00014-93-1-0743



ETU GUALLE LA LA LA

Ronald A. Roy

The Applied Physics Laboratory University of Washington 1013 NE 40th St. Seattle, WA 98105

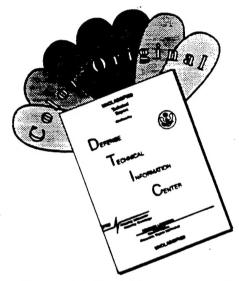
William M. Carey

Advanced Research Projects Agency 3701 North Fairfax Drive Arlington, VA 22203

January 22, 1996

19960202 027

# DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

# LOW-FREQUENCY RESONANCE SCATTERING FROM ACOUSTICALLY COMPACT BUBBLE CLOUDS

A Report to the ONR Under the Support of Grant N00014-93-1-0743

Ronald A. Roy

The Applied Physics Laboratory University of Washington 1013 NE 40th St. Seattle, WA 98105

William M. Carey

Advanced Research Projects Agency 3701 North Fairfax Drive Arlington, VA 22203

January 22, 1996

#### ABSTRACT

Recent developments indicate that microbubble plumes and layers are produced when waves break and are convected to depth. The fundamental question is: what role, if any, is played by these microbubble plumes in the production of sound near the sea surface in the low (20 Hz) to mid (2 kHz) frequency range? Measurements of ocean ambient noise levels show a dramatic increase in mid-frequency ambient noise when wave breaking occurs. Furthermore, measurements of sound scattering from the sea surface has a different characteristic than predicted by Bragg scattering from gravity waves, i.e. there exists a large zero-Doppler component. We hypothesized that if microbubble clouds and plumes with void fractions greater than 10-4 act as collective resonant oscillators, then radiated noise can be produced and scattering can occur with little Doppler shift but ample Doppler spread. This hypothesis was based on the theory that when a large number of microbubbles with individual resonance frequencies far above the frequency of excitation are present in an acoustically compact region, the mixture properties determine the radiation and scattering. We present scattering measurements from submerged bubble clouds which show that at the lower frequencies the resonance effect is significant, with the lowest order mode being a monopole which can be modeled as the volumetric pulsation of an acoustically compact sphere. Moreover, when such a cloud is located near the sea surface, a dipole radiation pattern results. Consequently, the scattered intensity is a sensitive function of depth, frequency, surface roughness and grazing angle. We show that for certain sea state conditions, observable anomalous effects can result from transient plumes generated close to the surface.

#### TABLE OF CONTENTS

1.	Preface	4	
3.			
5.			
7.	Papers in refereed journals and edited conference proceedings		
	-		
	Invited presentations		
	Contributed presentations		
8.	Appendix A:	R.A. Roy, W.M. Carey, M. Nicholas, J.A. Schindall and L.A. Crum, "Low-frequency scattering from submerged bubble clouds," J. Acoust. Soc. Am. 92, 2993-2996, 1992	
	Appendix B:	L.A. Crum, J.A. Schindall, R.A. Roy and W.M. Carey, "Low-frequency resonance backscatter from near-surface bubble clouds", SACLANTCEN Proc. of the NATO Symposium on Low-Frequency Active Sonar, Lerici, Italy, B9.1-9.16., 1993 B1-B16	
	Appendix C:	W.M. Carey and R.A. Roy, "Sound scattering from micro- bubble distributions near the sea surface," <u>Ocean Rever-</u> <u>beration</u> , ed. by D.D. Ellis, J.R. Preston and H.G. Urban, Kluwer Acad., Dordrecht, Netherlands, 25-44, 1993	
	Appendix D:	W.M. Carey, and R.A. Roy, "The low-frequency radiation and scattering of sound from bubbly mixtures near the sea surface,"  Proceedings of the 2nd European Conference on Underwater  Acoustics, ed. by L. Bjorno, Pub. by European Commission.,  Luxembourg, 207-212 (1994)	
	Appendix E:	R.A. Roy, J.A. Schindall, W.M. Carey, and L.A. Crum, "Can near-surface bubble clouds and plumes lead to anomalous perturbations in low-frequency sea-surface scattering?"  Proceedings of the 2nd European Conference on Underwater Acoustics, ed. by L. Bjorno, Pub. by European Commission, Luxembourg, 195-202 (1994)	
	Appendix F:	J.A. Schindall, "Low-Frequency Scattering from Submerged Bubble Clouds," Doctoral Thesis, University of Mississippi, December, 1995	

#### 1. PREFACE

This Document constitutes a Final Report for ONR Grant N00014-93-1-0743, "The Acoustical Properties of Dense Bubble Clouds". This effort focused on the physical aspects of low-frequency acoustic scattering from densely populated bubble clouds, and was funded under Code 321 Ocean acoustics (M. Orr, M. Badiey, and J. Simmen). The report is both chronological and archival, with the detailed technical descriptions relayed by way of a series of Appendices, each containing separate publications detailing the ongoing evolution of the project.

In order to orient the reader a background introduction to the key scientific issues is presented, followed by a brief description of the contents of each report appearing in the appendices, an assessment of future work, a list of references and a description of project personnel. A summary of the external communications exercised over the course of the project is then presented, which leads into the detailed technical descriptions given in the Appendices. For the most comprehensive technical treatment, the reader is directed to Appendix F, which contains the text of a Doctoral Thesis written by J. Schindall, the primary graduate student working on the project (under the supplementary assistance of an ONR AASERT Grant N00014-93-1-0749).

#### 2. PROJECT BACKGROUND

Recent studies indicate that microbubble layers, plumes and clouds are produced when waves break (w.s. > 6m/sec), and that these are initially taken to depth by the breaking wave downward orbital momentum and thereafter by convection and Langmuir circulation [Thorpe, 1982; Su et. al., 1984; Monahan and MacNiocaill, 1986; Monahan and Lu, 1990]. A fundamental question to be answered is what role, if any, do microbubble assemblages play in the near-surface production and scattering of sound in the low to mid-frequency range (20 Hz to 2 kHz). There exists ample experimental evidence to suggest that mid-frequency ambient noise levels dramatically increase when waves begin to break [Carey and Browning, 1988; Kennedy and Goodnow, 1990; Carey and Fitzgerald, 1993; Hollett, 1994] and that sound scattering from the sea surface possesses a characteristic which is different from that expected by Bragg scattering from gravity waves and that target-like returns from the second convergence zone [McDaniel, 1988; Carey et al., 1989]. The commonly held view was that persistent bubble clouds, plumes or layers brought on by breaking wave activity can act as efficient scatterers of sound. Indeed, several published works [Henyey, 1991; McDonald, 1991; Gragg and Wurmser, 1993; Carey and Roy, 1993; Gilbert, 1993; Sarkar and Prosperetti, 1993] indicate a significant role for bubble assemblages in surface backscatter.

In response to these observations a hypothesis has been stated that if microbubble clouds and plumes with void fractions (B) ranging from 10-3 to 10-2 are entrained by surface wave activity, then low-frequency radiated noise can be produced and scattering can occur with little Doppler shift and ample Doppler spread. It has been argued by Prosperetti [1988a&b] and by Carey and Browning [1988] that, at low frequencies, it is not the individual bubbles driven at their natural resonance frequencies that contribute the most to the scattering cross section of a bubble plume. Rather, sound is scattered primarily by the bubble plume itself via a process in which the bubbles pulsate together in a collective mode of oscillation. This hypothesis is based on the classical theory that when a large number of small bubbles occupy an acoustically compact region coupled oscillations result, and the propagation of sound is determined by the properties of the mixture; the free gas establishes compressibility and the water provides inertia. In such a case, a damped resonant oscillation can result in which the eigenfrequency is determined by the phase speed within the "effective" medium (typically less that 500 m/sec) and the characteristic length scale of the plume. There exists ample evidence of this phenomena, which was originally theoretically proposed by Reschevkin [1963] and Morse and Ingard [1968], first applied to the dynamics of bubble clouds formed by hydrodynamic cavitation by d'Agostino and Brennen [1983, 1988], and to bubble clouds in the sea by Carey and Browning [1988] and by Prosperetti [1988a,b]. Experimental results have been obtained from the study of low-frequency noise produced by steady-state bubble columns [Yoon et al., 1991; Kozhevnikova and Bjorno, 1991; Roy et al., 1991a&b; Nicholas et al., 1994] as well as transient bubble plumes in a tank by Kolaini et al., [1993, 1994] and in the free field by Carey et al. [1993].

The Office of Naval Research under the direction of M. Orr formulated a Sea Surface Scattering Special Research Project in the fall of 1988 to determine the dominant scattering mechanisms at the sea surface. At that time scattering from extended near surface distributions and bubble plumes and clouds from breaking wave were considered. It is interesting to note that at the initiation of this SRP, a workshop in Feb. of 1989 placed fairly accurate estimates of the contributions from each mechanism. To further resolve these issues, W. Carey, A. Prosperetti and L. Crum found themselves among a small group of investigators advocating laboratory and lake experiments designed to illuminate the mechanisms governing low-frequency sound scattering from bubble clouds and plumes. From this core interaction a lake experiment evolved. The experiment, performed in September 1990, was conducted at ≈24m depth at Lake Seneca, NY by investigators from NUSC with investigators from the NCPA, UCONN, Drexel, with support from BBN and Kildare. The scattering experiment, which was

designed principally by W. Carey and R. Roy, proved partially successful; a large backscatter effect was observed at low frequencies, however, uncertainties in bubble cloud characteristics and questions concerning multipath reverberation, parametric bubble excitation and individual-bubble resonance scattering confounded attempts to quantitatively reconcile this observation with theory.

Since the burden of proof lies with the experimentalists, we decided to repeat the scattering experiment at Lake Seneca, this time at a deeper depth (93 m) and using both conventional and parametric sources accompanied by *in situ* video measurements of bubble size distribution as well as bubble cloud geometry, orientation and rise velocity. The rational behind the experiment was to measure the frequency-dependent backscatter target strength of a submerged bubble cloud of known properties in the free field under known propagation conditions. The working hypothesis is a follows: is the low-frequency scattering cross section for a bubble cloud (β order 10-3) determined by the bubble sizes and number density (as in the case of coherent or incoherent resonance and off-resonance scattering from individual bubbles), or is it governed by the free-gas volume fraction and the length scales for the cloud proper? In this experimental effort we did not set out to duplicate "realistic" salt-water bubble clouds, but rather to obtain data to test and validate theory in order that ocean experiments could be more effectively conceived and designed and a mechanism for target like returns from the sea surface. A secondary goal was to test various means of producing the sound beams and bubble clouds necessary for such experiments.

#### 3. REPORT SUMMARY

As indicated earlier, the technical content of this Report is relayed by way of a series of published articles that describe the status of the project at its various stages of development. To get the most comprehensive and up-to-date description of the work performed under this Grant, the reader is referred to Appendix F. Below I give a capsule description of the contents of each Appendix in order to orient the reader.

This is the first paper to come out of the project effort. Published as a Letter in the Journal of the Acoustical Society of America, the paper gives a brief summary of the preliminary results from the second Lake Seneca experiment (Lake Seneca-II), as well as a

Appendix A: R.A. Roy, W.M. Carey, M. Nicholas, J.A. Schindall and L.A. Crum, "Low-frequency scattering from submerged bubble clouds," J. Acoust. Soc. Am. 92, 2993-2996, 1992.

capsule description of the details of the test range and the methods employed. The data presented in this paper were obtained on-line during the test, and has not been post processed. Similarly, reported calculations of predicted bubble cloud scattering properties are based on preliminary video records of cloud releases and have undergone some improvement over the course of the grant period. Nevertheless, the underlying physics of the process are clearly indicated. There exists a pronounced low-frequency resonance peak in the backscatter target strength suggestive of *bubble-cloud* (as opposed to single-bubble) monopole resonance scattering. This peak is remarkably large for a bubble cloud so small (TS order 0 dB, cloud size order 0.5 m). Moreover, the results obtained with parametric and conventional sound sources were found to be comparable for frequencies ranging from 500 Hz to 3 kHz (with a seemingly anomalous discrepancy at 1 kHz). This helps put to rest the controversy surrounding the use of a parametric sound source to do bubble scattering experiments, at least for the conditions encountered in this experiment.

Appendix B: L.A. Crum, J.A. Schindall, R.A. Roy and W.M. Carey, "Low-frequency resonance backscatter from near-surface bubble clouds", SACLANTCEN Proceedings of the NATO Symposium on Low-Frequency Active Sonar, Lerici, Italy, B9.1-9.16., 1993.

The fundamental conclusion of the ASA Letter in Appendix-A is that three-dimensional bubble clouds possess a monopole scattering characteristic that scales with the colume of the cloud and the void fraction of free gas; the latter establishing an effective sound speed within the cloud. Working within this conceptual framework, it is logical to consider such a cloud to be acoustically compact, and therefore approximate to a point scatterer. In Appendix B, we present an article that takes the acoustically compact bubble cloud is brings it close to the sea surface, modeled as a pressure release plane. The backscatter target strength is computed using the method of images and results suggest that high void-fraction clouds resident (however briefly) near the sea surface might be the cause of short-lived scattering features ("spiky returns") observed at higher sea states.

This paper, derived from an invited presentation at a NATO symposium on ocean reverberation held at the SACLANT Center, outlines the theoretical construction governing low-frequency bubble cloud scattering. Based on the classical works of Foldy [1945],

Appendix C: W.M. Carey and R.A. Roy, "Sound scattering from micro-bubble distributions near the sea surface," <u>Ocean Reverberation</u>, ed. by D.D. Ellis, J.R. Preston and H.G. Urban, Kluwer Acad., Dordrecht, Netherlands, 25-44, 1993.

Reshevkin [1963], and Morse and Ingard [1968], this paper addresses in a systematic fashion the problem of scattering from bubbly assemblages of increasing complexity, starting with incoherent scattering from individual bubbles at low void fraction, to multiple scattering from bubbly structures at moderate void fraction, and culminating with resonance bubble cloud scattering at high void fractions occupying acoustically compact spaces. The paper concludes with a brief synopsis of the Lake Seneca-II experiment - similar to the one that appears in Appendix-A.

Appendix D: W.M. Carey, and R.A. Roy, "The low-frequency radiation and scattering of sound from bubbly mixtures near the sea surface," <u>Proceedings: The 2nd European Conference on Underwater Acoustics</u>, ed. by L. Bjorno, Pub. by European Commission, Luxembourg, 207-212 (1994).

This is a companion paper to Appendix-D. It sketches the theoretical framework for bubble cloud scattering at both low and high void fractions, focusing primarily on the latter. The problem of acoustic backscatter from a compliant sphere is addressed and the asymptotic expression appropriate to low frequencies is presented. Comparison between the long-wavelength theory and the results from the Lake Seneca scattering and "bucket drop" [Carey et al., 1994] experiments are made. Features of both the low-frequency transient emissions associated with energetic air entrainment as well as the low-frequency backscatter from compact bubble clouds are semi-quantitatively described by the long-wavelength theory.

Appendix E: R.A. Roy, J.A. Schindall, W.M. Carey, and L.A. Crum, "Can near-surface bubble clouds and plumes lead to anomalous perturbations in low-frequency seasurface scattering?" <a href="Proceedings: The 2nd European Conference on Underwater Acoustics">Proceedings: The 2nd European Conference on Underwater Acoustics</a>, ed. by L. Bjorno, Pub. by European Commission, Luxembourg, 195-202 (1994).

This paper builds on the foundation laid by Crum et al. (see Appendix-B), addressing the problem of low-frequency acoustic backscattering from near-surface bubble clouds modeled as monopole scatterers near a planar surface. The solution to the problem of scattering from a compliant sphere (presented in Appendix-E) is coupled to an effective medium approximation to yield the backscatter target strength for an acoustically compact, spherical bubble cloud. In addition, updated experimental results coming out of Lake Seneca-II (monopole resonance frequency and target strength at resonance) exhibit quantitative agreement with the theoretical predictions. This supports the contention that highly compressible, acoustically compact scatters with low aspect ratio can indeed be modeled as

spheres in as much as monopole scattering is concerned: the target strength is primarily dependent on scatterer volume. To bring such a scatterer to the surface, we employ the method of images coupled to a the Kirchoff approximation for the Pierson-Moskowitz sea surface and for a specific set of boundary layer conditions [Thorsos, 1990] to determine the frequency- and windspeed-dependent refection coefficient. Results suggest that bubble clouds with depths on the order of 1-2 m, length scales on the order of 20 cm and void fraction on the order of 0.001 can generate target strengths as high a -5 dB for frequencies ranging from 500 Hz to 1 kHz. This is in qualitative agreement with the images of backscatter from near surface (< 5 m) bubble clouds obtained by Adair *et al.* [1992].

Appendix F: J.A. Schindall, "Low-Frequency Scattering from Submerged Bubble Clouds," Doctoral Thesis, University of Mississippi, December, 1995.

This is the Doctoral Thesis of Jeffrey Schindall, the principal graduate student working the project. This is the most comprehensive and archieval reference available on the Lake Seneca-II experiment and the subsequent data analysis. The motivation, rationale, and chronology of the effort is presented, along with a detailed description of the experimental methods and the data analysis, a summary of the results obtained at low-to-moderate frequencies (250 Hz - 3 kHz) and a comparision of the experimental results with both the complete and asymptotic (low frequency) theories for scattering from a compliant sphere. The Dissertation also describes the procedures employed to model the backscattering from the bubble cloud (treated as a monopole scatterer) as it is brought near a model see surface. (This is essentially the work previously reported in Appendix E.) Conclusions are drawn and suggestions for future work presented.

#### 4. FUTURE WORK

Even with the advent of J. Schindall's dissertation, the analysis of the results from Lake Seneca experiment is not yet complete. There continue to be facets to the data that defy explanation. Most prominent are several peaks in the calculated backscattered target strength versus frequencies above 700 Hz that do not present themselves in the data. When one considers the complex shape of the rising bubble cloud (a blunt bullet with a skirt of bubbles), our failure to resolve the higher-order modes of the sphere comes as no surprise. Unfortunately, we lack the detailed knowledge about the cloud shape and internal distribution of void fraction required to generate a higher-frequency model. All we know about the cloud is it's approximate volume and average void fraction, and that's all we require to predict the monopole resonance frequency and target strength - a cornerstone contribution of this study.

More mystifying are measured undulations in the measured target strength as a function of bubble cloud position as it rises through the beam of the parametric source. These fluctuations are suggestive of interference between the cloud and the bubbler, a notion that has yet to be rigorously explored. Finally, the Q of the monopole resonance peak is much broader than it should be. We believe that poor signal to noise at off-resonance scattering frequencies may contribute to measured target strengths biased in the upward direction by broadband noise and reverberation. Both of these issues will be brought to the front burner, and will serve as the focus of ongoing analysis on the part of the authors of this report.

#### 5. REFERENCES

- Adair, R., Huster, M., and Miklovic, D. (1992) J. Acoust. Soc. Am. 92, 2479.
- Carey, W.M. and Browning, M. (1988) "Low frequency ocean ambient noise measurement and theory," <u>Sea Surface Sound</u>, ed. by B.R. Kerman, Kluwer Press, 361-376.
- Carey, W.M., Goodman, R., McDaniel, S., Jackson, D., Deferrari, H., and Brisco, M. (1989) "Sea surface scatter experiments, A review," Memo to M. Orr, ONR (11250A), 17 August.
- Carey, W.M. and Roy, R.A. (1993) "Sound scattering from microbubble distributions near the sea surface,"

  Ocean Reverberation, ed. by D.D. Ellis, J.R. Preston and H.G. Urban, Kluwer Acad., Dordrecht,
  Netherlands, 25-44.
- Carey, W. M. and Fitzgerald, J. W. (1993) "Low-frequency noise from breaking waves," <u>Natural Physical Sources of Underwater Sound</u>, ed. by B. Kerman, Kluwer Acad. Pub., Dordrecht, Netherlands, 277-304.
- Carey, W. M., Fitzgerald, J. W., Monahan, E. C., and Wang, Q. (1993) "Measurement of the sound produced by a tipping trough with fresh and salt water," J. Acoust. Soc. Am. 93, 3178-3192.
- d'Agostino and Brennen (1983) "On the acoustical dynamics of bubble clouds," ASME Cavitation and Multiphase Flow Forum, 72-75.
- d'Agostino, L. and Brennen, C.E. (1988) "Acoustical absorption and scattering cross sections of spherical bubble clouds", J. Acoust. Soc. Am. 84, 2126-2134.
- Foldy, L. (1945) "Multiple scattering of waves," Phys. Rev. 67, 107-119.
- Gilbert, K.E. (1993) "A stochastic model for scattering from the near surface oceanic bubble layer," J. Acoust. Soc. Am. 94, 3325-3334.
- Gragg, R.F. and Wurmser, D. (1993) "Low-frequency scattering from intermediate bubble plumes: Theory and computational parameter study", J. Acoust. Soc. Am. 94, 319-329.
- Henyey, F. (1991) "Acoustic scattering from ocean microbubble plumes in the 100 Hz to 2 kHz region," J. Acoust. Soc. Am. 90, 399-405.
- Hollett, R.D. (1994) "Observations of underwater sound at frequencies below 1500 Hz from breaking waves at sea," J. Acoust. Soc. Am. 95, 165-170.
- Kennedy, R.M. and Goodnow, T. (1990) "Measuring the vertical directional spectra caused by sea surface sound," JOE Oct. 90, 299-310.
- Kolaini, A.R., Roy, R.A., Crum, L.A., and Mao, Y. (1993) "Low-frequency underwater sound generated by impacting transient cylindrical water jets," J. Acoust. Soc. Am. 94, 2809-2820.
- Kolaini, A.R., Roy, R.A., and Gardner, D.L. (1994) "Low-frequency acoustic emissions in fresh and salt water," J. Acoust. Soc. Am. 96 (3), 1766-1772.
- Kozhevnikova, I. N., Bjorno, L. (1991) "Transformation of signals from high to low frequency in a bubble screen." in <u>Conference Proceedings: Ultrasonics International 1991</u>, Butterworth, 591-595.
- McDaniel, S. (1988) "High-frequency sea surface scattering: Recent Progress," J. Acoust. Soc. Am. 84(S1), S121.
- McDonald, B. E. (1991) "Echoes from vertically striated subresonant bubble clouds: A model for ocean surface reverberation," J. Acoust. Soc. Am. 89, 617-622.

- Monahan, E. C. and MacNiocaill, G. (eds.) (1986) Oceanic Whitecaps and Their Role in Air-Sea Exchange Processes, D. Reidel, Boston.
- Monahan, E. C. and Lu, M. (1990) "Acoustically relevant bubble assemblages and their dependence on meteorological parameters," IEEE Jour. Ocean Engr. 15, 340-349.
- Morse, P. and Ingard, U.K. (1968) Theoretical Acoustics, McGraw-Hill, NY, 422-428.
- Nicholas, M., Roy, R.A., Crum, L.A., Oguz, H.N. and Prosperetti, A. (1994) "Sound emission from a laboratory bubble cloud," J. Acoust. Soc. Am. 95 (6), 3171-3182.
- Prosperetti, A. (1988a) "Bubble dynamics in ocean ambient noise," in <u>Sea Surface Sound</u>, ed. by B. R. Kerman, Kulwer Acad., Boston, MA, 151-171.
- Prosperetti, A. (1988b) "Bubble related ambient noise in the ocean," J. Acoust. Soc. Am. 84, pp. 1042-1054.
- Rschevkin, S.N. (1963) The Theory of Sound, Pergamon Press, NY, 370-374.
- Roy, R.A., Yoon, S.W., Nicholas, M., Park, K.J., and Crum, L.A. (1991) "Collective oscillations of a bubble cloud as a source of underwater ambient noise in the ocean," <a href="Proceedings: Internoise 91">Proceedings: Internoise 91</a>, ed. by A. Lawrence, Sydney, Australia, 399-402.
- Roy, R.A., Nicholas, M., Markiewicz, K. and Crum, L.A. (1991) "Collective oscillations in fresh and salt water bubble clouds," J. Acoust. Soc. Am. 90, 2318.
- Sarkar, K. and Prosperetti, A. (1993) "Backscattering of underwater noise by bubble clouds," J. Acoust. Soc. Am. 93, 3128-3138.
- Su, M. Y., Green, A. W., and Bergin, N. T. (1984) <u>Gas Transfer at Water Surfaces</u>, ed. by W. Brutsaert and G. Jirka, Reidel Press, 211-219.
- Thorpe, S. A. (1982) "On the clouds of bubbles formed by breaking wind-waves in deep water and their role in air-sea gas transfer," Phil. Trans. Roy. Soc. A 304, 155-185.
- Thorsos, E. (1990) "Acoustic scattering from a 'Pierson-Moskowitz' sea surface," J. Acoust. Soc. Am. 88, 335-249.
- Yoon, S. W., Crum, L. A., Prosperetti, A., and Lu, N. Q. (1991) "An investigation of the collective oscillations of a bubble cloud," J. Acoust. Soc. Am. 89, 700-706.

#### 6. PROJECT PERSONNEL & SPONSORS

The experiment was conducted by the principal investigators R. Roy (NCPA) and W. Carey (NUWC) and their respective teams. NUWC (W. Konrad, T. Stottlemeyer, Seneca Barge Facility Crew) was responsible for sources, receivers, range installation and calibration. NCPA (M. Nicholas, J. Schindall, L. Crum, J. Burch) provided the submerged bubbler and video system, assisted in the installation of the test range, and handled the data acquisition and analysis. UCONN (M. Wilson, E. Monahan) was responsible for certain aspects of the bubble size measurement and analysis. BBN (W. Marshall) provided low-frequency conventional sources. Although the test truly represented a joint effort, the bulk of the data acquisition and analysis was performed by Roy, Carey, Nicholas, Schindall and Crum.

Finally, the sponsors for this effort must be congratulated for their faith in our ability to perform this multi-institutional effort. The NCPA group was supported by the ONR (Ocean Acoustics, M. Orr and AEAS, R. Fedden). NUSC was supported by ONR (M. Orr), NUSC IR (K. Lima), and the ONT (C. Votaw, T. Goldsberry). Continuing data and theoretical analysis was carried out by R. Roy and J. Schindall at APL/UW with support provided by ONR (M. Badiey, J. Simmen). The document constitutes the final report for the latter effort.

### 7. EXTERNAL COMMUNICATIONS RESULTING FROM THIS GRANT (partial support indicated by asterisk)

#### Papers Published in Refereed Journals and Edited Conference Proceedings:

- \*R.A. Roy, S.W. Yoon, M. Nicholas, K.J. Park and L.A. Crum, "Collective oscillations of a bubble cloud as a source of underwater ambient noise in the ocean," <u>Proceedings: Internoise 91</u>, ed. by A. Lawrence, Sydney, Australia, 399-402, 1991.
- \*S.W. Yoon, K.J. Park, L.A. Crum and R.A. Roy, "Acoustic backscattering from a cylindrical bubble cloud in water," <u>Technical Papers of the Western Pacific Acoustics Conference IV</u>, The Australian Acoustical Society, Brisbane, Australia, 462-469, 1991.
- R.A. Roy, W. Carey, M. Nicholas and L.A. Crum, "Low-frequency scattering from resonant bubble clouds," Proceedings 14th ICA 2, C3-5, 1992.
- R.A. Roy, W. Carey, M. Nicholas, J. Schindall and L.A. Crum, "Low-frequency scattering from submerged bubble clouds," J. Acoust. Soc. Am. 92(5), 2993-2996, 1992.
- L.A. Crum, R.A. Roy and A.P. Prosperetti, "The underwater sounds of precipitation," Naval Research Review 44(2), 2-12, 1992.
- W. Carey and R.A. Roy, "Sound scattering from microbubble distributions near the sea surface," <u>Ocean Reverberation</u>, ed. by D.D. Ellis, J.R. Preston and H.G. Urban, Kluwer Acad., Dordrecht, Netherlands, 25-44, 1993.
- \*A.R. Kolaini, R.A. Roy, L.A. Crum and Y. Mao, "Low-frequency underwater sound generated by impacting transient cylindrical water jets," J. Acoust. Soc. Am. 94(5), 2809-2820, 1993.
- R.A. Roy and L.A. Crum, "Bubble-related sources of ocean ambient noise," <u>Proceedings: Ultrasonics</u> International 93, Butterworth Heinemann Ltd., Oxford, UK, 1993.
- L.A. Crum, J.A. Schindall, R.A. Roy and W.M. Carey, "Low-frequency resonance backscatter from near-surface bubble clouds", SACLANTCEN Proceedings of the NATO Symposium on Low Frequency Active Sonar, Lerici, Italy, B9.1-9.16. (1993).
- \*M. Nicholas, R.A. Roy, L.A. Crum, H.N. Oguz and A. Prosperetti, "Sound emission from a laboratory bubble cloud," J. Acoust. Soc. Am. 95 (6), 3171-3182 (1994).
- \*A.R. Kolaini, R.A. Roy and D.L. Gardner, "Low-frequency acoustic emissions in fresh and salt water," J. Acoust. Soc. Am. 96 (3), 1766-1772 (1994).
- R.A. Roy, J.A. Schindall, W.M. Carey, and L.A. Crum, "Can near-surface bubble clouds and plumes lead to anomalous perturbations in low-frequency sea-surface scattering?" <u>Proceedings of the 2nd European</u> <u>Conference on Underwater Acoustics</u>, ed. by L. Bjorno, Pub. by European Commission, Luxembourg, 195-202 (1994).
- W.M. Carey, and R.A. Roy, "The low-frequency radiation and scattering of sound from bubbly mixtures near the sea surface," <u>Proceedings of the 2nd European Conference on Underwater Acoustics</u>, ed. by L. Bjorno, Pub. by European Commission, Luxembourg, 207-212 (1994).

- \*S.A. Cheyne, C.T. Stebbings and R.A. Roy, "Phase velocity measurements in bubbly liquids using a fiber optic laser interferometer," J. Acoust. Soc. Am. 97, 1621-1624 (1995).
- \*P.A. Hwang, R.A. Roy, and L.A. Crum, "Artificial bubble cloud targets," <u>Sea Surface Sound (3)</u>, ed. by M.J. Buckingham and J.R. Potter, Kluwer Acad. Pub., Dordrecht, Netherlands, in press.
- \*P.A. Hwang, R.A. Roy and L.A. Crum, "Artificial bubble cloud targets for underwater acoustic remote sensing, J. Atmos. and Ocean Tech., accepted for publication.

#### Papers Read Before Professional Organizations: Invited Presentations

- \*L.A. Crum, R.A. Roy, S.W. Yoon, A.R. Kolaini and M. Nicholas, "Bubble-related sources of sea surface sound," presented at the special session on *Acoustical Oceanography* at the 119<sup>th</sup> meeting of the Acoustical Society of America, San Diego, CA, November 1990.
- R.A. Roy, "The mechanical properties of bubbles, bubbly liquids, and bubble clouds," Colloquium presented at the Department of Mechanical Engineering, Johns Hopkins Univ., Baltimore, MD, January 1991.
- R.A. Roy, W. Carey, M. Nicholas, L.A. Crum and J. Schindall, "Low-frequency backscatter from dense submerged bubble clouds," presented at the special session on Acoustical Studies of Upper Ocean Processes at the 123<sup>rd</sup> meeting of the Acoustical Society of America, Salt Lake City, UT, May 1992.
- \*L.A. Crum, R.A. Roy, M. Nicholas and A. Kolaini, "Laboratory experiments on bubble cloud oscillations," presented at the special session on *Acoustical Studies of Upper Ocean Processes* at the 123<sup>rd</sup> meeting of the Acoustical Society of America, Salt Lake City, UT, May 1992.
- W.M. Carey, R.A. Roy and R.R. Goodman, "Surface scattering from microbubble distributions near the sear surface," presented at the SACLANT Ocean Reverberation Symposium, La Spezia, Italy, May 1992.
- R.A. Roy, W.M. Carey, M. Nicholas and L.A. Crum, "Low-frequency scattering from resonant bubble clouds," presented at the meeting of the 14t<sup>th</sup> ICA, Beijing, China, September 1992.

#### Papers Read Before Professional Organizations: Contributed Presentations

- M.S. Korman and R.A. Roy, "The scattering of sound by a cylindrical bubble cloud," presented at the 118<sup>th</sup> meeting of the Acoustical Society of America, St. Louis, MO, November 1989.
- \*A.R Kolaini, M. Yi, R.A. Roy and L.A. Crum, "The acoustic signatures of laboratory-generated bubble plumes," presented at the 120<sup>th</sup> meeting of the Acoustical Society of America, San Diego, CA, November 1990.
- \*M. Nicholas, R.A. Roy and L.A. Crum, "Collective oscillations of a bubble column: higher modes," presented at the 120<sup>th</sup> meeting of the Acoustical Society of America, San Diego, CA, November 1990.
- \*A.R. Kolaini, R.A. Roy and L.A. Crum, "The hydrodynamic and acoustic behavior of bubble plumes generated by an impacting water jet," presented at the 121st meeting of the Acoustical Society of America, Baltimore, MD, April 1991.
- R.A. Roy, M. Nicholas and L.A. Crum, "Low-frequency backscattering from a submerged bubble cloud," presented at the 121<sup>st</sup> meeting of the Acoustical Society of America, Baltimore, MD, April 1991.

- \*S.A. Cheyne and R.A. Roy, "Sound velocity measurements in a bubble cloud," presented at the 122<sup>nd</sup> meeting of the Acoustical Society of America, Houston, TX, November 1991.
- \*R.A. Roy, M. Nicholas, K. Markiewicz and L.A. Crum, "Collective oscillations in fresh and salt water bubble clouds," presented at the 122<sup>nd</sup> meeting of the Acoustical Society of America, Houston, TX, November 1991.
- R.A. Roy, L.A. Crum, M. Nicholas, J. Schindall, W.A. Carey, W.A. Konrad, W.J. Marshall, E.C. Monahan and A.P. Prosperetti, "Low-frequency acoustic scattering from a submerged bubble cloud," presented at the 122<sup>nd</sup> meeting of the Acoustical Society of America, Houston, TX, November 1991.
- \*R.A. Roy, S.W. Yoon, M. Nicholas, K.J. Park and L.A. Crum, "Collective oscillations of a bubble cloud as a source of underwater ambient noise in the ocean," presented at Internoise 91, Sydney, Australia, November 1991.
- \*S.W. Yoon, K.J. Park, L.A. Crum and R.A. Roy, "Acoustic backscattering from a cylindrical bubble cloud in water," presented at the Western Pacific Acoustic Conference IV, Sydney, Australia, November 1991.
- \*A.R. Kolaini, C.C. Church, R.A. Roy and L.A. Crum, "The acoustic emissions from bubble plumes generated by an impacting transient water jet," presented at the meeting of the 14t<sup>th</sup> ICA, Beijing, China, September 1992.
- J. Schindall, W.M. Carey, R.A. Roy, L.A. Crum and M. Nicholas, "Sound scattering from near-surface bubble clouds," presented at the 124<sup>th</sup> meeting of the Acoustical Society of America, New Orleans, LA, October 1992.
- \*S.A. Cheyne, C.T. Stebbings and R.A. Roy, "Acoustic phase velocity measurements in a bubbly liquid using a fiber optic laser Doppler velocimeter," presented at the 124<sup>th</sup> meeting of the Acoustical Society of America, New Orleans, LA, October 1992.
- L.A. Crum, J.A. Schindall, R.A. Roy and W.M. Carey, "Low-frequency resonance backscatter from near-surface bubble clouds," presented at the Low-Frequency Active Sonar Symposium, SACLANT Undersea Research Center, La Spezia, Italy, May 1993.
- R.A. Roy and L.A. Crum, "Bubble-related sources of ocean ambient noise," presented at Ultrasonics International 93, Vienna, Austria, July 1993.
- R.A. Roy, "Low-frequency acoustic scattering from bubble assemblages near the ocean surface," presented at the IUTAM Symposium on Bubble Dynamics and Interface Phenomena, Birmingham, England, September 1993.
- J.A. Schindall, R.A. Roy, L.A. Crum and W.M. Carey, "Low frequency backscatter from near-surface bubble plumes," presented at the 126th Meeting of the Acoustical Society of America, Denver, CO, October 1993.
- \*A.R. Kolaini, R.A. Roy and D.L. Gardner, "Low-frequency acoustic emissions from impacting transient cylindrical water jets in salt and fresh water," presented at the 126th Meeting of the Acoustical Society of America, Denver, CO, October 1993.
- \*P.A. Hwang, R.A. Roy, and L.A. Crum, "Artificial bubble cloud targets," presented at Sea Surface Sound '94, Lake Arrowhead, CA, March 1994.

- \*P.A. Hwang, R.A. Roy, and L.A. Crum, "Artificial bubble cloud targets for acoustic remote sensing of the ocean," presented at the 127th Meeting of the Acoustical Society of America, Cambridge, MA, June 1994.
- R.A. Roy, J.A. Schindall, W.M. Carey, and L.A. Crum, "Can near-surface bubble clouds and plumes lead to anomalous perturbations in low-frequency sea-surface scattering?" presented at the 2nd European Conference on Underwater Acoustics, Lyngby, Denmark, July 1994.
- W.M. Carey, and R.A. Roy, "The low-frequency radiation and scattering of sound from bubbly mixtures near the sea surface," presented at the 2nd European Conference on Underwater Acoustics, Lyngby, Denmark, July 1994.
- \*C. T. Stebbings, S. A. Cheyne and R. A. Roy, "Phase Velocity Measurements in Bubbly Liquids Using a Fiber Optic Laser Interferometer," presented at the Meeting of the Southeastern Section of the American Physical Society, Newport News, VA, November 1994.

### Appendix A

#### Low-frequency scattering from submerged bubble clouds

Ronald A. Roya)

Applied Physics Laboratory, University of Washington, 1013 NE 40th Street, Seattle, Washington 98105

William Careyb)

Defense Advanced Research Projects Agency, 79 Whippoorwill Road, Old Lyme, Connecticut 06371

Michael Nicholas, Jeffrey Schindall, and Lawrence A. Crum
National Center for Physical Acoustics, University of Mississippi, Coliseum Drive, University,
Mississippi 38677

(Received 18 February 1992; accepted for publication 15 July 1992)

Preliminary results are presented from a recent experiment carried out at the NUWC Seneca Lake test facility and designed to investigate the scattering properties of bubble clouds produced in fresh water in the absence of boundaries and under known propagation conditions. The test range consisted of both conventional and parametric sources, conventional receivers, and a transient bubbler that was submerged to 91 m. The cloud possessed an elongated teardrop shape (length≈1.4 m, diameter≈0.5 m) with a void fraction of 0.25% and a mean bubble radius of 1.5 mm, which corresponds to a single-bubble resonance frequency of 6.6 kHz at depth. Backscatter target strengths were measured for frequencies ranging from 300 Hz to 14 kHz. These measurements revealed high target strengths (up to + 1 dB) and distinctive peaks in the spectrum at 450 Hz, and 1.3, 3.0, 6.5, and 11 kHz. The low-frequency results are consistent with calculations (based on the theory of collective bubble oscillations) of the resonance frequency and target strength of a comparably sized, spherical bubble cloud. Certain features of the high-frequency backscatter data are consistent with single-bubble resonance scattering. These preliminary results are consistent with the notion that, at low frequencies, the scattering characteristics of a bubble cloud are determined, not by the bubble sizes and number density, but rather by the free-gas volume fraction and the length scales of the cloud proper.

PACS numbers: 43.30.Ft, 43.30.Gv

#### INTRODUCTION

Studies indicate that microbubble layers and bubble plumes are produced when waves break, and are convected to depth by Langmuir circulation. <sup>1,2</sup> A fundamental question to be answered is what role do microbubble layers and plumes play in the near-surface production and scattering of sound in the low- to mid-frequency range (20 Hz to 2 kHz). There exists ample evidence to suggest that sound scattering from the sea surface possesses a characteristic which is different from that expected by Bragg scattering from gravity waves.<sup>3</sup> It is our contention that breaking waves results in bubble clouds or plumes that can efficiently scatter sound. Indeed, recently published works<sup>4,5</sup> indicate a significant role for bubble plumes in surface backscatter.

At issue is the scattering cross section for a bubble plume. Is it determined by the bubble sizes and number density or is it related to the free-gas volume fraction and the length scales for the cloud proper? Prosperetti<sup>6,7</sup> and Carey et al.<sup>8,9</sup> maintain that low-frequency sound is scattered primarily by the bubble plume via a process in which the bubbles pulsate collectively. In such a case, a damped resonant oscillation can result in which the eigenfrequency is determined by the phase speed within the "effective" medium (typically less than 100 m/s) and the characteristic length scale of the plume. Ample evidence of this phenomenon has

been obtained from the study of steady-state bubble columns<sup>10</sup> and bubble screens.<sup>11</sup>

This Letter reports preliminary results from a experiment designed to measure the frequency-dependent back-scattering from a bubble cloud in fresh water in the absence of boundaries and under known propagation conditions. In this effort we did not set out to duplicate "realistic" salt-water clouds, but rather to obtain data to test and validate low-frequency scattering theory in order that ocean experiments could be more effectively conceived and designed. A more complete study is currently underway.

#### I. DESCRIPTION OF THE EXPERIMENT

The Seneca Lake, NY facility consists of two moored barges in a water depth of 130 m, with the smaller barge (10.7 m×42.7 m) serving as the platform for our test range. Equipment was deployed with the use of davits, a cable meter, and the edge of the barge, with the resulting vertical geometry shown in Fig. 1. Transmitters and receivers were oriented collinearly, with the axis of the range intersecting the path of a rising bubble cloud. The backscatter target strength for the cloud was determined over frequencies ranging from 300 Hz to 14 kHz using both conventional and parametric sources. The parametric source was positioned at a depth of 61.0 m and driven with a 22-kHz carrier signal that was up and down shifted by 1/2 the desired difference

a) Author to whom correspondence should be sent.

h) Work performed while at NUWC, New London, CT 06320.

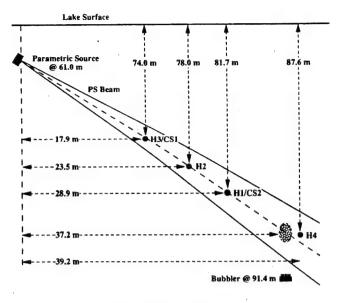


FIG. 1. Layout of the test range.

frequency, time-gated, and delivered to a 20-kW drive amplifier. Over the course of our measurements, the repetition frequency was fixed at either 1 or 2 Hz and the pulse lengths were made to vary from 5 to 15 ms. Calibrations of both the frequency-dependent source level and beamwidth were performed in situ and were nominally 177 dB ( $re: 1 \mu Pa$  at 1 m) and 8.0°, respectively (at 1.0 kHz).

The parametric source offered the advantage of low-frequency directionality. However, bubbly liquids are highly nonlinear and there is therefore a possibility of an "apparent" elevation in the measured target strength due to parametric excitation. In response to this, three conventional sources were also utilized. In the 500-Hz to 1.5-kHz range, we employed a Honeywell HX-29 (CS1 in Fig. 1). At 3 and 5 kHz, we used an ITC 6-in. spherical ball transmitter (CS2). At 5 kHz and above, we drove the parametric array with conventional (i.e., single-frequency) tone bursts. We ran experiments 87.6-m depth to minimize reverberation and multipath propagation.

In addition to these sources, the test range consisted of four ITC 6-in. spherical balls, acting as receivers and situated as indicated in Fig. 1 (H1-H4). The outputs of these hydrophones were low-pass filtered, 100-Hz high-pass filtered, preamplified and recorded. Sound velocities were measured daily. The near-surface sound speed profile (less than 8-m depth) was variable as one might expect. At depths > 70 m isovelocity conditions prevailed, with a sound speed of 1421.7 m/s at 70.0 m and 1420.8 m/s at 91.4 m. For all the calculations presented herein, a sound speed of 1421.5 m/s was used. The range was calibrated by using the 22-kHz primary from the parametric source to measure the time-offlight differences between hydrophones and thus verify alignment and slant ranges. We then inserted a 1.12-m-diam, hollow, steel sphere in order to test our ability to measure target strength. The target was positioned at the expected location of the bubble cloud as it crosses the axis of the range. Using the standard definition of target strength 12 and assuming a perfectly reflecting target, this sphere possessed a theoretical TS of - 11.1 dB at 5 kHz. Our measured TS was - 12.3 dB.

Repeatable bubble clouds were produced using a pressurized steel enclosure submerged to 91.4 m and vented to the lake via an array of hypodermic needles connected to a bank of solenoid valves; the air pressure and solenoid-valve operation were controlled from the surface. The array contained forty-eight 22-gauge hypodermic needles arranged in three concentric rings of 6.25-, 12.50-, and 18.75-cm radii. A differential pressure transducer, mass-flow meter, and underwater video camera enabled us to monitor the overpressure, the total volume of air expelled into a given cloud, and the characteristic dimensions of the cloud. A 3.5-s burst of air at an overpressure of 10 psi resulted in a roughly cylindrically shaped cloud (length  $\approx$  1.4 m, diameter  $\approx$  0.5 m) with a void fraction of 0.25%.

The bubble size distribution is a matter of critical importance, for it helps to distinguish single-bubble resonance scattering from multiple-bubble collective effects. By rigging the camera  $\approx 2$  cm from a needletip we obtained video images of bubbles from which we estimated the *in situ* distribution of bubble sizes at depth (see Fig. 2). This distribution peaks at about 1.55-mm radius, which corresponds to a reasonance frequency of  $\approx 6.6$  kHz at a depth of 87.6 m. Even the largest bubbles ( < 3 mm) possessed resonance frequencies of order 3.4 kHz. For frequencies less than  $\approx 2$  kHz, single-bubble resonance scattering can be neglected.

#### II. EXPERIMENTAL PROCEDURE

Target strength measurements were performed in the following manner. Once the range was aligned and calibrated and the source frequency and pulse duration determined, the solenoids were activated for 3.5 s, resulting in a rising bubble cloud. The cloud rise velocity was approximately 0.3 m/s, which yielded a time-varying backscattered echo level. The echo level evolved as the cloud traversed the beam, reaching a maximum level as the cloud crossed the beam axis. We monitored the evolution of the echo level by digitizing the echo return, storing subsequent returns in memory, and displaying a time-compressed view of the sequence of echoes produced by the passage of a single cloud, as shown in Fig. 3. The echoes were range-gated to remove the incident

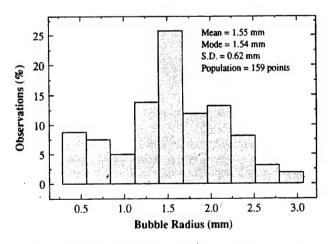
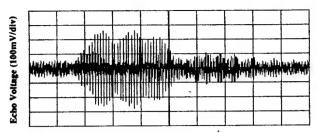


FIG. 2. Bubble-size distribution measured at a depth of 91.4 m.



Time Elapsed from the Generation of the Bubble Cloud (5sec/div)

FIG. 3. Typical scattering results from hydrophone H1, obtained with the parametric source operating at a 3.0-kHz difference frequency. This time-compressed view of 100 echo returns serves to illustrate the evolution of the echo level as the cloud rises through the source beam. The target strength for this cloud was -1.2 dB.

wave and to minimize reverberation. Note that the first maximum in Fig. 3 occurs at approximately the 13-s mark, which corresponds to the amount of time it takes the cloud to rise 3.8 m into the axis of the beam. The secondary echoes may be the consequence of angle-dependent scattering and are a subject of ongoing study.

The echo corresponding to the peak response was chosen for the determination of target strength, which we computed by measuring the peak voltage of the main pulse (MV) and the echo (EV) and inserting these quantities, along with the transmission loss factor (TLF), into the following expression for the target strength (TS):

$$TS = 20 \log[EV/MV] - TLF;$$

$$TLF = TL_{sr} - TL_{sr} - TL_{rr},$$
(1)

where the TLF accounts for the various transmission losses assuming spherical spreading of the propagating waves. The target strengths presented below consist of either single measurements or the mean values calculated from 2 to 3 repeated measurements. Error bars cannot be rigorously defined at this point due to the lack of statistical information regarding the precision of the calculated mean values; however, an ad hoc comparison of repeated measurements indicate fluctuations in the measured target strengths which ranged from  $\pm$  0.5 to  $\pm$  4 dB.

#### III. EXPERIMENTAL RESULTS

Target strengths in the mid- to high-frequency range are plotted in Fig. 4. The measured target strengths are quite large—recall that a 1-m-diam perfectly reflecting sphere has a target strength of — 12 dB at 5 kHz. There is evidence of resonance scattering peaks at 1.3, 3.0, 6.5, and possibly 11 kHz. From the bubble-size distribution, one can argue that the peak in the TS curve at about 6.5 kHz is due to single-bubble resonance scattering. However, the distribution does little to account for the other peaks, which are a focus of ongoing study.

Shown in Fig. 5 are the measured low-frequency target strengths; the parametric source data were obtained at two distinct tests run during July and September of 1991. Once again the target strengths are remarkably high; at 450 Hz, the measured TS is about 10 dB greater that one would expect from a 1-m-diam perfectly reflecting sphere. There are

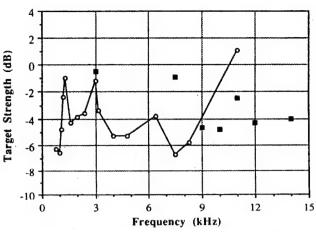


FIG. 4. Backscatter target strengths measured in the mid- to high-frequency range. The solid squares were conventional-source measurements obtained by driving the parametric source with a single primary. The open circles were obtained using the parametric source. The solid line is added to help illustrate the trends.

apparent resonance peaks at 1.3 kHz and 450 Hz. For these peaks to be the result of single-bubble resonance scattering the bubbles would have to possess radii of 7.9 mm and 2.3 cm, respectively, which is highly unlikely.

It is probable that these peaks are the result of resonance scattering due to collective oscillations of the bubble cloud. The fundamental resonance for a spherical cloud is given approximately by<sup>9</sup>

$$f_0 \approx \frac{1}{2\pi a} \left( \frac{3\gamma P_0}{\chi (1-\chi)\rho} \right)^{1/2},\tag{2}$$

where  $\gamma$ ,  $P_0$ ,  $\rho$ , and a are, respectively, the polytropic exponent, the ambient pressure, the water density, and the effective radius of the cloud treated as a sphere. The void fraction  $\chi$  is the ratio of the volume of free gas within the cloud to the total volume of the cloud. [Equation (2) holds for  $0.002 < \chi < 0.94$ .] A cloud of void fraction 0.0025 and "effective" radius 0.5 m (1/2 the average of the cloud diameter and the cloud length) possesses a calculated resonance fre-

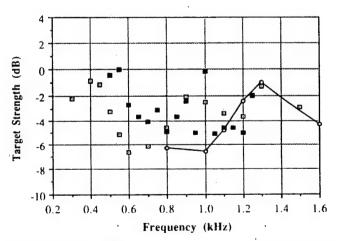


FIG. 5. Backscatter target strength measured in the low- to mid-frequency range. The solid squares were obtained using the HX-29 conventional source positioned as shown in Fig. 1. The open circles and dotted squares were the results of parametric-source measurements run in July 1991 and October 1991, respectively.

quency of approximately 405 Hz. Moreover, the target strength is given by 12

$$TS = 10 \log [I_s/I_i]_{r=1 \text{ m}}, \tag{3}$$

where  $I_s$  and  $I_i$  are the scattered and incident intensities, respectively. For the bubble cloud oscillating collectively, this is given approximately by

$$\frac{I_s}{I_i} \approx \frac{z_0^6}{9(kr)^2} \frac{\{1 - (x/x')\}^2}{[1 - (x/x')z_0^2/3]^2 + [(x/x')z_0^3/3]^2},$$

$$r = 1 \text{ m.}$$
(4)

where  $z_0 = ka$ ,  $x = \rho c^2$ , and  $x' = \rho_m c_m^2$ , where the subscript m denotes the effective properties of the air/water mixture. (We use the definition of the void fraction to obtain  $\rho_m$  and Wood's result<sup>13</sup> to determine  $c_m$ .) Since Eq. (4) does not account for thermal damping, this result is only accurate provided the damping decrement is of order  $3 \times 10^{-3}$  or less. For a 0.5-m-radius cloud with a void fraction of 0.0025, we calculate a target strength of approximately -6 dB at 405 Hz. These values for the resonance frequency and target strength of the cloud exhibit qualitative agreement with our measurements in the 400-500 Hz range.

A comparison between the low-frequency measurements obtained with the parametric and conventional sources indicate reasonably good agreement. Although reverberation made it difficult to obtain conventional source data for frequencies less than 1 kHz, data obtained in the 500-Hz to 3-kHz range using the parametric source, the HX-29, and the ITC 6-in. sphere were comparable. However, parametric and conventional source measurements at 1, 7, and above 9 kHz differed by as much as 5 dB. (Our confidence in the 1-kHz conventional source measurement is low, due to the large spread in the data at this frequency. An analysis of all the data obtained at this frequency has yet to be performed.) It appears that enhanced parametric excitation did not appreciably bias the target strength measurements for frequencies below about 9 kHz.

#### IV. CONCLUSIONS

Measurements of backscatter TS versus frequency have been made using conventional and parametric sources. The results using these two types of sources were found to be comparable for frequencies ranging from 500 Hz to 3 kHz (the 1-kHz data not withstanding). Backscatter TS were quite high, exceeding that of a comparably sized, perfectly reflecting sphere by as much as 15 dB. Certain features of the high-frequency backscatter data are consistent with single-bubble resonance scattering while measurements at the low frequencies ( < 1 kHz) appear to be governed by the collec-

tive oscillations of the cloud. It is evident that several features of the backscatter spectra require additional analysis.

In summary, we conclude that our measured target strengths reflect both single and collective bubble behavior, with the latter playing the more prominent role at the lower frequencies.

#### **ACKNOWLEDGMENTS**

The authors would like to thank Martin Wilson of the Univ. of Connecticut as well as Joe Burch, Paul Dakin, and Ali Mehrabi of NCPA for their lending their expertise to this effort. We would especially like to thank William Konrad of NUWC and William Marshall of BBN for their expert assistance with the parametric and conventional sources, and Andrea Prosperetti of Johns Hopkins Univ. for his insight and advice. Finally, we acknowledge the financial support of Marshall Orr, ONR; Ken Lima, NUWC IR; as well as AEAS, ONT, and DARPA.

<sup>1</sup>S. A. Thorpe, "On the clouds of bubbles formed by breaking wind-waves in deep water and their role in air-sea gas transfer," Philos. Trans. R. Soc. London Ser. A 304, 155-185 (1982).

<sup>2</sup>E. C. Monahan and M. Lu, "Acoustically relevant bubble assemblages and their dependence on meteorological parameters," IEEE J. Ocean Eng. **15**(4), 340-349 (1990).

<sup>3</sup>S. McDaniel, "High-frequency sea surface scattering: Recent progress," J. Acoust. Soc. Am. Suppl. 1 84, S121 (1988).

<sup>4</sup>B. E. McDonald, "Echoes from vertically striated subresonant bubble clouds: A model for ocean surface reverberation," J. Acoust. Soc. Am. 89, 617–622 (1991).

<sup>5</sup>F. Henyey, "Acoustic scattering from ocean microbubble plumes in the 100 Hz to 2 kHz region," J. Acoust. Soc. Am. 90, 399-405 (1991).

<sup>6</sup>A. Prosperetti, "Bubble related ambient noise in the ocean," J. Acoust. Soc. Am. 84, 1042-1054 (1988).

<sup>7</sup>A. Prosperetti, "Bubble dynamics in ocean ambient noise," in *Sea Surface Sound*, edited by B. R. Kerman (Kulwer Academic, Boston, MA, 1988), pp. 151-171.

\*W. M. Carey and D. G. Browning, "Low-frequency ocean ambient noise: Measurements and theory," in Ref. 7, pp. 361-376.

<sup>9</sup>W. M. Carey, J. W. Fitzgerald, and D. G. Browning, "Low-frequency noise from breaking waves," in *Natural Physical Sources of Underwater Sound*, edited by B. R. Kerman (Kluwer Academic, Dordrecht, Netherlands, 1990). [Also: NUSC Tech. Rep. 8783, 5 Oct. 1990, avail. DTIC #AD227969.]

<sup>10</sup>S. W. Yoon, L. A. Crum, A. Prosperetti, and N. Q. Lu, "An investigation of the collective oscillations of a bubble cloud," J. Acoust. Soc. Am. 89, 700-706 (1991).

<sup>11</sup>I. N. Kozhevnikova and L. Bjorno, "Transformation of signals from high to low frequency in a bubble screen," in *Conference Proceedings: Ultrasonics International 1991* (Butterworth, London, 1991), pp. 591-595.

<sup>12</sup>R. J. Urick, Principles of Underwater Sound (McGraw-Hill, New York, 1975), pp. 263-278.

<sup>13</sup>A. B. Wood, A Textbook of Sound (Bell, London, 1932/1955), pp. 360-

## Appendix B

In Ocean Reverberation, ed. by D.D. Ellis, J.R. Preston and H.G. Urban, Kluwer Acad., Dordrecht, Netherlands, 25-44, 1993.

SOUND SCATTERING FROM MICROBUBBLE DISTRIBUTIONS NEAR THE SEA SURFACE

William M. Carey
Defense Advanced Research Projects Agency
3701 N. Fairfax Drive
Arlington, VA 22203-1714

Ronald A. Roy Applied Physics Laboratory University of Washington 1013 NE 40th Street Seattle, WA 98105

ABSTRACT. Backscatter from the sea surface is thought to be governed by the roughness of the surface and subsurface bubble distributions. At low frequencies, due to the paucity of large bubbles, scattering results primarily from coherent and/or collective scatter from bubbles entrained by the subsurface vorticity or carried to depth by the Langmuir circulation and thermal convection. It is shown that scattering from compact regions is a function of the volume fraction of air and to first order can be described by a Minnaert formula modified with the volume fraction. Measurement of sound scattering from a submerged cloud of bubbles produces low-frequency peaks with large low-frequency target strength consistent with this theory.

#### 1. INTRODUCTION

Recent experimental evidence shows that when waves break, microbubble layers and plumes are produced, and these bubbly assemblages are convected to depth by the Langmuir circulation as well as the thermal convection found in the oceanic mixed layer [1,2,3,4]. The fundamental question to be answered is what role, if any, do microbubble layers and plumes play in the near-surface radiation and scattering of sound in the low- to mid-frequency range (20 Hz to 2 kHz). exists ample evidence that sound scattering at these frequencies has a different characteristic than expected from Bragg scattering from gravity waves [5,6]. It is our hypothesis that if these microbubble clouds and plumes have structures with volume fractions exceeding 10 then these structures can radiate as well as scatter sound through a dampened resonant oscillation [6,9,10,11,12,13]. Recent analytical studies of low-frequency sound scattering from geometrically shaped bubble distribution beneath the sea surface [7,8] indicate that these types of bubble distributions may also coherently scatter sound.

This paper discusses these issues in terms of the classical theories of sound scattering from complex but compact distributions of bubbles. In particular, our intention is to show, with the Morse and

Ingard formulation [19], the relationship between the Born approximation [19], continuum theory, and single-bubble incoherent scattering. We show that when the bubbles are treated as scatters in the Lighthill method that one proceeds from single-bubble scattering, to coherent scattering, and finally to the scattering from the distribution as if it were a continuum. In fact, the continuum theory is shown to provide a bound on the region of validity of the Born approximation. The problem is also scaled by the volume fraction and its effect on the liquid and gas mixture compressibility. Finally, we present the results of a free bubble cloud scattering experiment conducted at Lake Seneca. These measurements clearly illustrate the complex nature of the scattering cross section of a compact bubble cloud.

#### 2. A REVIEW OF SCATTERING OF SOUND FROM COMPACT BUBBLE DISTRIBUTIONS

The classical theory of sound radiation and scattering from compact regions containing bubbly liquids is a mature subject and several texts [14-21] contain very useful overviews. In recent years, the scattering of sound from bubble distributions near the sea surface has received renewed interest[6]. This is in part due to the work of Thorpe [1] and his observation of subsurface bubble plumes and clouds. McDaniel [5] has shown that high frequency (>1 kHz) sea surface scattering can be explained using bubble distributions near the sea surface coupled with composite rough surface scattering. However, the bubble size distributions [4,5] measured at sea are too small to explain scattering at frequencies less than 1 kHz using a single bubble scattering formalism. The purpose here is to discuss this lowfrequency bubble scattering from bubble plumes and clouds in light of classical scattering theories as well as to place current analytical work [7,8,9,10,11,12,13] in perspective. Consequently, we closely follow and use material found in these texts, most notably the Morse and Ingard text [19].

Fig. 1 shows the geometry for a compact bubble cloud. The primary focus is on distributions of salt water bubbles with radii less than 1 mm, resonant frequencies greater than 3 kHz, and excitation frequencies less than 1 kHz. Since we are treating a compact source, the sea surface will be incorporated using the method of images.

The Fourier transform of the wave equation is

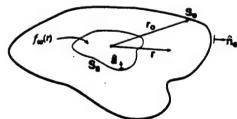


Figure 1. The geometry of the general scattering problem

$$\nabla^2 P_{\omega}(r) + k^2 P_{\omega}(r) = -F_{\omega}(r) , \qquad (1)$$

where  $F_{\omega}(r)$  represents a general source distribution. Since the problem is a farfield radiation one and since our source region is compact, the more general retardation notation has not been used. The Green's equation is

$$\nabla^2 g_{\omega}(r|r_o) + k^2 g_{\omega}(r|r_o) = -\delta (\vec{r} - \vec{r}_o). \tag{2}$$

where  $g_{\omega}(r/r_o) = \exp(i\vec{k}\cdot(\vec{r}-r_o))/4\pi|\vec{r}-\vec{r}_o|$  and will be used in this form throughout this paper. The standard procedure is to multiply 1) by  $g_{\omega}$ , 2) by  $P_{\omega}$ , subtract, use reciprocity to interchange r and  $r_o$ , and then use the divergence theorem to yield:

$$P_{\omega}(r) = \iiint g_{\omega}(\vec{r}/\vec{r}_{o}) F_{\omega}(r_{o}) dV_{o} + \iiint \left[ g_{\omega} \frac{\partial P_{\omega}}{\partial n_{o}} - P_{\omega} \frac{\partial g_{\omega}}{\partial n_{o}} \right] ds_{o} . \tag{3}$$

This equation describes both the radiation and scattering of sound from a region V<sub>o</sub>. The surface integral in the radiation problem goes to zero as r<sub>o</sub> increases due to imposition of the radiation condition. This integral in the case of the scattering problem becomes the incident wave coming in from infinity. Thus both the radiation and scattering of sound require the evaluation of the volume integral, the difference being the source of excitation. In this paper we will simply use a plane wave, consequently

$$P_{\omega}(r) = P_{i\omega}(r) + \iiint g_{\omega}(\vec{r}|\vec{r}_o) P_{i\omega}(r_o) f_{\omega}(r_o) dr_o$$

$$F_{\omega}(r) = P_{i\omega}(r_o) f_{\omega}(r_o), P_{i\omega}(r) = P_o e^{i\vec{k}_o \cdot \vec{r}}$$
(4)

The various approaches to scattering can be considered as consisting of the method by which  $f_{\omega}(r_{\circ})$  is determined coupled with the means by which the volume integral can be solved. Traditional treatments employed at higher frequencies consider bubble densities, the number of bubbles per wavelength volume, are small and multiple scattering is therefore not a problem. However, individual bubble resonance is important, thus:

$$f_{\omega}(r_o) = f_n(\omega_o, \theta, r_n) \delta(\vec{r} - \vec{r}_n) , \qquad (5)$$

where  $f_n(\omega_o,\theta,r_n)$  represents the response function of the nth bubble. Substitution of this expression for  $f_\omega$ , and after some algebraic manipulation, we obtain for a uniform distribution of N bubbles the following expression for the scattered intensity

$$I_{s}(r) = \frac{|P_{s\omega}|^{2}}{2\rho c} = \frac{P_{o}^{2}}{2\rho cr^{2}} |f(\omega_{o}, \theta)|^{2} \left[N + \sum_{m \neq n} \exp(i\vec{\mu} \cdot \vec{R}_{mn})\right], \tag{6}$$

where  $\vec{\mu} = \vec{k}_o - \vec{k}$ ,  $|\mu| = 2k \sin(\theta/2)$ . When the summation within the brackets is zero, the scattered intensity is seen to be proportional to the number of bubbles N; i.e., the incoherent sum of scattering from individual bubbles. For the case of an acoustically compact spherical volume of radius a, the bracketed terms can be shown to be [18]

$$I_s(r) - N^2 \left[ 1 - \frac{2}{5} \frac{(N-1)}{N} k^2 a^2 \sin^2(\theta/2) \cdots \right].$$
 (7)

This is the coherent scattering case where the scattered intensity is directional and proportional to  $N^2$ . The distinguishing factor between these two expressions is the number of bubble within  $V_{\circ}$  or the volume fraction.

The incoherent, proportional to N, scattering particularly applies to non-compact distributions of bubbles which are sparse. This is the case treated by McDaniel [5] and has been shown to describe several sea surface scattering measurements at frequencies >1 kHz and at low grazing angles.

The work of Foldy [14,16] was based on the above case of  $f_{\omega}(r)$ , except it included the effects of scattering from neighboring bubbles

(here it is assumed that they are mono-dispersed in radius with resonance frequency  $\omega_0$ ):

$$f_{\omega_n}(r_o) = P_{\omega_n}(r_n) f_n(\omega_o, \theta_n, r_n) \delta(\vec{r} - \vec{r}_n) \quad \text{and}$$
 (8)

$$P_{\omega}(r) = P_{i\omega}(r) + \sum_{n=1}^{N} \frac{e^{ikr_n}}{r_n} P_{\omega n}(r_n) f_n. \tag{9}$$

The problem when the interaction between bubbles must be considered is the choice of  $P_{\omega n}(r_n)$ . Foldy [16] recognized that this quantity depends on the incident field and the sum of the scattered field from all the bubbles and could be written as

$$P_{\omega n} = P_{i\omega}(r_n) + \sum_{m \neq n}^{N} \frac{e^{ikr_{mn}}}{r_{mn}} f_m P_{\omega m}(r_n) = P_{\omega}(r_n) - \frac{f_n}{r_n} P_{\omega n}(r_n) e^{ikr_n}. \tag{10}$$

This result when substituted in equation 8 yields

$$P_{\omega}(r) = P_{i\omega}(r) + \sum_{n=1}^{N} P_{\omega}(r_n) f_n \frac{e^{ikr_n}}{r_n}, \qquad (11)$$

$$\xrightarrow{N\to\infty} P_{i\omega}(r) + \int P_{\omega}(r_n) f_n \frac{e^{ikr_n}}{r_n} dr_n.$$

Foldy then recognized that the configurational average of equation (9) is equivalent to the solution of

$$\nabla^{2} \langle P_{\omega}(x) \rangle + k_{s}^{2} \langle P_{\omega}(x) \rangle = 0, \quad k_{s}^{2} = k^{2} + 4\pi \langle f_{n} \rangle, \tag{12}$$

which is a wave equation with a different wave number or index of refraction. This index of refraction is determined by the continuum properties of the bubbly volume [11,22,23].

The concept that sound scattering from a cloud of air bubbles in water is described as a progression from incoherent, coherent to continuum scattering was fully developed in references [14-21] prior to the mid-nineteen sixties. Morse and Ingard [19] have provided an extension of these results based on the Lighthill method.

The use of Lighthill's approach requires the determination of  $f_{\omega}(r_{\circ})$  from the two-component conservation equations. Retaining only second order terms and neglecting the Lighthill stress tensor yields

$$f_{\omega}(r_o) = k^2 \gamma_k P_{\omega}(r_o) - \vec{\nabla}_o (\gamma_o \vec{\nabla}_o P_{\omega}(r_o)) , \qquad (13)$$

where  $\gamma_{\kappa}$  and  $\gamma_{\rho}$  are the compressibility and density contrasts given by

$$\gamma_k = \frac{\kappa_e - \kappa}{\kappa}, \gamma_\rho = \frac{\rho_e - \rho}{\rho}$$
 for  $r_o < a$  and  $\gamma_k = \gamma_\rho = 0$  for  $r_o > a$ . (14)

Substitution of this form into equation (4) yields

$$P_{\omega}(r) = P_{i\omega}(r) + \iiint [k^2 \gamma_{\kappa} P_{\omega} g_{\omega} - \vec{\nabla}_{o} \cdot (\gamma_{\rho} \vec{\nabla}_{o} P_{\omega}) g_{\omega}] dV_{o}.$$
 (15)

This equation is extremely interesting, because substitution of the mean compressibility contrast  $(\overline{\gamma}_{\kappa})$  and mean density contrast  $(\overline{\gamma}_{\rho})$  results in a wave equation that is similar to Foldy's with an effective

wavenumber given by  $[1+\overline{\gamma}_{\rho}]\overline{\nabla_{o}^{2}P_{\omega}(r_{o})}+k^{2}(1-\overline{\gamma}_{\kappa})P_{\omega}(r_{o})$  where  $k^{2}(1-\overline{\gamma}_{\kappa})/(1-\overline{\gamma}_{\rho})$  is the effective wavenumber squared. This is another way of showing that the region may be described by its mixture density and compressibility. This also provides a means by which the scattering regimes may be distinguished. To show this we integrate equation 15 by parts and use our free space Green's function:

$$P_{\omega}(r) = P_{o}e^{i\vec{k}_{o}\vec{r}} + \frac{P_{o}e^{ik_{s}r}}{r}\phi(k_{s}), \text{ where}$$

$$\phi(k_{s}) = \frac{k^{2}}{4\pi P_{o}}\iiint \left[\gamma_{\kappa}P_{\omega} - i\gamma_{\rho}\frac{\vec{k}_{s}}{k^{2}}\nabla_{o}P_{\omega}\right]e^{-i\vec{k}_{s}\vec{r}_{o}}dV_{o}.$$
(16)

This quantity is referred to as the angle distribution factor and is recognized as a three-dimensional transform of the compressibility and density contrasts. Since Fourier methods apply, the function  $\phi(\vec{k}_s) = F(\phi(\vec{r}_o))$  can also be determined by the convolution of the spatial transforms of  $P_\omega$  and the  $\gamma$ 's. Since the scattered intensity is proportional to  $|\phi(k_s)|^2$ , the full statistical description of the power spectral density and correlation properties can thus be used. This is an important observation, for the description of non-compact near-surface bubble layers can be treated in this manner. Gilbert [24] has employed these techniques to characterize low-frequency sound scattering from subsurface bubble distributions. The case of a compact distribution is presented here.

The problem is still to specify  $P_{\omega}$  within the volume integral [15]. The method of successive approximations has been used to solve this equation. This method approximates the term  $P_{\omega}$  by a sum of successively higher order terms  $P_{\omega}(r_o) = P_{i\omega}(r_o) + P_1 + P_2 + \cdots$ . When  $P_{i\omega}(r_o) > P_1$  throughout the source volume, we simply allow  $P_{\omega}(r_o) = P_{i\omega}(r_o)$ . This approximation is referred to as the Born approximation. This approach has been applied to bubble scattering in underwater acoustics for over 30 years. When we assume the internal field to be given by the incident field, the function  $\phi_s(k_s,k_o)$  reduces to (Morse and Ingard [19]):

$$\phi(k_s, k_o) = \frac{k^2}{4} \pi \iiint [\gamma_k + \gamma_\rho \cos\theta] e^{-i\vec{\mu} \cdot \vec{r}_o} dV_o$$

$$= 2\pi^2 k^2 [\Gamma_\kappa(\vec{\mu}) + \Gamma_\rho(\vec{\mu}) \cos\theta].$$
(17)

This expression, when applied to a sphere of radius a, yields the well-known result [19]:

$$\phi_{\sigma}(\vec{\mu}) = 1/3k^2a^3\left(\langle \frac{\kappa_{\sigma}-\kappa}{\kappa} \rangle + \langle \frac{\rho_{\sigma}-\rho}{\rho} \rangle \cos\theta\right)$$
 (18)

This result yields a scattered intensity proportional to the number of bubbles squared, equivalent to our previous result of coherent scattering.

This result from the use of the classical Born approximation can be shown to be limited to the treatment of the bubbly liquid in  $V_o$  as a continuum with different  $\rho c^2$ . To show this result we return to our schematic (Fig. 1) and treat the source volume as being bounded by a surface integral,  $S_o$ , the region of r would thus be between  $S_o$  and the surface at distance  $S_o$ . For the inner surface  $S_o$ ,  $\vec{a}$  is the outward directed normal into volume  $V_o$ . The resulting pressure field is

$$P_{\omega}(r) = P_{i\omega}(r) + \iint \left[ P_{\omega} \frac{\partial g_{\omega}}{\partial a} - g_{\omega} \frac{\partial p}{\partial a} \right] dS_{a}. \tag{19}$$

The radiation and scattering problems are now reduced to specifying  $P_{\omega}/(\partial P/\partial a)$  on the surface  $S_{a}$  bounding the medium with properties  $\rho_{a}$ ,  $C_{a}$ . The material in the sphere will determine this ratio, as this is equivalent of requiring continuity of velocity and pressure at the boundary. The procedure is to use spherical harmonics to expand the field within the sphere, the field external to the sphere, and the incident plane wave (Anderson [21], also Carey [12]). The mathematical manipulation is straightforward but tedious. Morse and Ingard show that to the lowest order in ka the result is

$$\Phi_s = 1/3k^2a^3\left(\frac{\kappa_m - \kappa}{\kappa} + \frac{3\rho_m - 3\rho}{2\rho_m - \rho}\cos\theta\right). \tag{20}$$

This expression allows us to define the region of applicability of the Born approximation and the beginning of the region determined by the volume fraction in which the bubbly liquid should be treated as a continuum. The case of air bubbles in water when  $\kappa_e > \kappa$  and  $\rho_e \sim \rho$  we find

$$\Phi - (1/3) k^2 a^3 \left( \frac{\rho c^2}{\rho_m c_m^2} - 1 \right) - 1/3 k^2 a^3 \left( \frac{c^2}{c_m^2} - 1 \right). \tag{21}$$

This theoretical result, which, we stress, is taken from the classical literature, has attempted to place the question of sound scattering from compact bubble distributions in context. In particular it was our intention to show the relationship between the Born approximation, continuum theory, and single incoherent bubble scattering. The validity of these approximations was shown to depend on the quantity  $(c^2/c_m^2-1)$ , which is proportional to  $\chi$ , the volume fraction of the free air in the water volume.

The discussion has focused on a sphere of radius a. We have used this geometry because of its simplicity, but our conclusions for both the radiation and scattering from a complex region do not depend on the geometry. For an arbitrary complex yet compact distribution of air bubbles in water the radiated field can be shown to consist of monopole, dipole, quadrupole, and higher order terms. In our application the dipole and higher terms are not important. The monopole term corresponds to the volume pulsation; hence this treatment of a spherical source should also apply to any complex distribution radiating or scattering in the volumetric mode. This is similar to the observations made by Strasberg concerning single-bubble radiation [25].

#### 3. SOUND PROPAGATION IN BUBBLY MIXTURES

In a bubbly mixture, in which the of bubbles possess resonance frequencies which are much greater than the frequencies of interest, then the propagation of sound in the mixture can be described by an effective wave equation with mixture density and sound speed. These effective wave equations have been studied by several investigators (i.e., Foldy [14,16], Crighton [26], Van Wijngaarden [24]) and confirmed by Karplus (1958) at frequencies <1 kHz (See Carey [11] as well as [12,13] and Prosperetti [9]; also Ruggles [29]).

A. B. Wood [22] showed that at low frequencies in an air-water mixture that when the size of the bubbles and spacing between bubbles

were small compared to a wavelength, that for a given volume fraction  $(\chi)$  the mixture density  $(\rho_{\tt m})$  and compressibility  $(\kappa_{\tt m})$  are under equilibrium conditions given by

$$\rho_m = (1 - \chi)\rho_1 + \chi \rho_g \text{ and } \kappa_m = (1 - \chi)\kappa_1 + \chi \kappa_g, \qquad (22)$$

where I represents the liquid and g the gaseous component. The low-frequency limiting sonic speed,  $C_{\rm alf}$ , is seen to be

$$C_{mlf}^2 = \frac{dP}{d\rho} = (\rho_m \kappa_m)^{-1}. \tag{23}$$

In the case of the air bubble-water mixture, the process can be considered either adiabatic or isothermal. Since the controlling factor is the rate of heat transfer during bubble compression to the surrounding fluid which is rapid in water due to its large thermal capacity, the bubble oscillations at low frequencies may be considered isothermal. The following equations result:

$$C_{mlf}^{-2} = \frac{(1-\chi)}{C^2} + \frac{\chi^2}{C_g^2} + (\chi) (1-\chi) \frac{\rho_g^2 C_g^2 + \rho^2 C^2}{\rho_g C_g \rho C}$$

$$\chi \to 0 \ C_m^{-2} \to C^{-2} \text{ and } \chi \to 1 \ C_m^{-2} \to C_g^{-2}$$
(24)

and when  $0.002 < \chi < 0.94$ ,  $C_{mlf}^2 = P/\rho \chi (1-\chi)$ .

These equations show the large effect on sonic speed as a function of void fraction and relative bulk modulus. In the low-frequency range (<1 kHz) Karplus used an acoustic tube to determine the standing wave pattern as a function of air volume fraction and verified these results. Ruggles (1987) [29] has extended Karplus's technique to study a wide range of volume fractions. This behavior is due to the mixture's mass being primarily due to the liquid while its compressibility is due to the gas. Since we are dealing with the low-frequency response, much less than any individual bubble resonant frequency, only the volume fraction is important and not the bubble size distribution.

This equation clearly shows the dispersive character of the mixture speed of sound. In Fig. 2, we show several curves for a cloud of bubbles with individual resonant frequencies near 9.18 kHz. Below the resonance range we see the low-frequency minimum predicted by Wood's equation and above the resonance range we see the curves approach the speed of sound determined by the liquid compressibility. The phase velocity reaches a minimum at the resonance frequency and goes supersonic for a range of frequencies slightly above f. This is due to the fact that when driven above resonance bubbles pulsate out of phase with the driving force. Thus the bubbles "appear" to be stiff and the phase velocity subsequently increases dramatically. These strong dispersive effects need to be considered when describing sea surface scattering.

Equation (24) does not show the dispersive nature of this sonic speed variation. If one considers a bubbly liquid with a uniform bubble size whose corresponding resonant frequency is  $\omega_o$  and dampening  $\delta(\omega)$ , then one [23,26] can solve the component conservation equations to obtain the following frequency-dependent sonic speed:

$$\frac{1}{C_m^2} = \frac{(1-\chi)^2}{C^2} + \frac{1}{C_{mlf}^2(1-\omega^2/\omega_o^2+2i\delta\omega/\omega_o)}.$$
 (25)

This expression reduces to our previous equation for  $C_{alf}^2$  when  $\omega < \omega_o$ . The equation does show a dispersive character as illustrated in Fig. 3, where we have used the explicit expression from Prosperetti [23]. The dispersive character of the phase velocity is the reason that volume fraction measurements are necessary to determine the low-frequency sound speed changes near the surface of the sea.

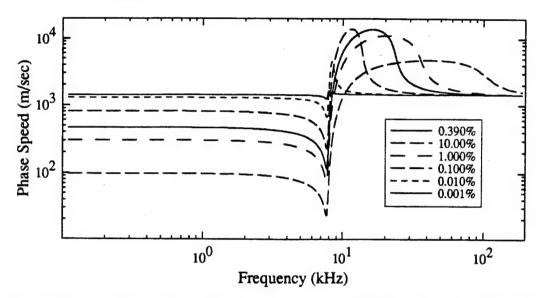


Figure 2. Dispersion curves for a cloud of 9.18 kHz bubbles

#### 4. THE OSCILLATION OF A BUBBLE SPHERICAL VOLUME

The generation and scattering of sound from a compliant sphere immersed in a fluid can be found in classical texts on the theory of acoustics (see Reschevkin, or Morse). In this particular case, the bubbly sphere has no well-defined boundary, but nonetheless is localized by perhaps a vortex or some other circulatory feature beneath the breaking wave. Since this analysis is somewhat standard [20,12,29], only a brief outline will be presented.

Here we assume that the bubbly region (Fig. 4) is compact with an arbitrary radius  $r_o$  and the region is composed of microbubbles with resonance frequencies far above the frequency of excitation. Buoyancy forces and restoring forces such as surface tension are not important. The properties of the bubbly region are described by the mixture speed  $c_a$  and density  $\rho_a$  with a resulting compressibility  $1/\rho c^2$ . The microbubbles supply the compressibility and the liquid supplies the inertia.

Fig. 3 shows the random collection of microbubbles within a radii r<sub>o</sub> from the origin, where ( $P_1$  = incoming wave or excitation,  $\rho_m$ ,  $c_m$  = the properties of the mixtures,  $\rho$ , c = the properties of water, and  $P_s$  = the scattered sound.)

P<sub>1</sub> ( , c ) , c

Figure 3. The random collection of micro bubbles.

The source of excitation is assumed to be global compared to the dimensions of the compact sphere. This assumption means we can consider a plane wave expanded in terms of spherical harmonics. The

physical reasoning is that the properties of the bubbly region determine its ability to radiate provided there is a source of excitation; i.e.,

$$P_{i} = P_{o} \exp(i\omega t - ikr) = P_{o} \exp(i\omega t) \sum_{m=0}^{\infty} i^{m} (2m+1) P_{m}(\theta) J_{m}(kr). \quad (26)$$

We require continuity of velocity and pressure at the generalized radius, r<sub>o</sub>. Furthermore, a radiation condition is imposed at large r and the field is required to remain finite within the bubbly region.

The particle velocity 
$$V_i = (-1/i\omega\rho) \partial P_i/\partial r$$
, (27)

Continuity of Pressure 
$$P_i(a) + P_s(a) = \overline{P}(a)$$
, (28)

Continuity of Velocity 
$$V_1(a) + V_2(a) = \overline{V_2}(a)$$
. (29)

The procedure is to assume the scattered wave or radiated wave is a sum of outward propagating spherical waves.

$$P_{s} = \exp(i\omega t) \sum_{m=0}^{\infty} A_{m} \overline{P}_{m}(\theta) H_{m}(kr) \exp(-i\epsilon_{m}kr).$$
 (30)

The pressure field inside the volume is expanded in terms of spherical Bessel function of the first kind; i.e.,

$$\overline{P} = \exp(i\omega t) \sum_{m=0}^{\infty} \overline{A}_m P_m(\theta) J_m(kr).$$
 (31)

When the region is compact  $\lambda > 2\pi r_o$ , we can solve for the coefficients  $A_m$  and  $\overline{A_m}$  by using the boundary conditions (27, 28, and 29) and equating each m order term. To  $O(k^3 r^3)$ , we find for  $A_o$  and  $A_i$  the following:

$$A_{0} = \frac{iP_{o}(k_{o}r_{o})^{3} (1 - y/\overline{y})/3}{\left[(1 - y/\overline{y} \frac{(kr_{o})^{2}}{3}\right] - i (kr_{o})\left[y/\overline{y} \frac{(kr_{o})^{2}}{3}\right]} \text{ and}$$
(32)

$$A_1 = \frac{P_0(kr_0)^3 (\rho_m - \rho)}{(2\rho_m + \rho)}, \tag{33}$$

where  $y=\rho c^2$  and  $\overline{y}=\rho_m c_m^2$ . The term A, becomes interesting when the real part of the denominator is equal to zero, we may say a monopole resonance has occurred. This occurs when

$$(kr_o)^2 = \left(\frac{2\pi f_o r_o}{C}\right)^2 = \frac{3\rho_m C_m^2}{\rho C^2} \text{ and } f_o = \frac{C}{2\pi r_o} \sqrt{\frac{3\rho_m C_m^2}{\rho C^2}},$$
 (34)

since 
$$p_m \sim p_e$$
 and  $C_m^2 \sim \gamma P/\rho_1 \chi (1-\chi)$ ,  $f_o = \frac{1}{2\pi r_o} \sqrt{\frac{3\gamma P}{\rho \chi}}$ , (35)

which we recognize as a modified Minnaert formula for the resonance frequency of a volume oscillation of a gas bubble (Carey and Fitzgerald [12]). (Note that we have inserted a factor  $\gamma$ , the ratio of specific heats. This factor applies to individual bubble pulsations and for the cloud we set  $\gamma=1$  corresponding to isothermal condi-

tions.) This result follows because the resonance angular frequency is proportional to the compressibility of the bubbly region, characterized by the stiffness  $(4\pi r_o \rho C^2)$  and the inertia  $(4\pi r_o^3 \rho/3)$ .

#### 5. THE MEASUREMENT OF BACKSCATTER FROM A SUBMERGED BUBBLE CLOUD

We report results from an experiment designed to measure low-frequency acoustic backscattering from a bubble cloud in fresh water under known propagation conditions. At issue is the scattering cross section for a bubble cloud. Is it determined by the bubble sizes and number density (as in the case of coherent or incoherent resonance and off-resonance scattering from individual bubbles) and discussed previously, or is it related to the free-gas volume fraction and the length scales for the cloud proper? In this effort we did not set out to duplicate "realistic" saltwater bubble plumes, but rather to obtain data to test bubble cloud scattering theories.

#### 5.1 Experimental Procedures

The Naval Undersea Warfare Center (NUWC) Seneca Lake, New York test facility consists of two moored barges in 130 m of water, with the smaller barge (10.7 m x 42.7 m) serving as the platform for our test range. Equipment was deployed with the use of davits, a cable meter, and the edge of the barge, with the resulting vertical geometry shown in Fig. 4. Parametric and conventional transmitters and conventional receivers were orientated colinearly, with the axis of the range intersecting the path of a rising bubble cloud. Though the actual test consisted of measurements for frequencies ranging from 300 Hz to 14 kHz, only the low-frequency (<1.6 kHz) results will be presented here.

The parametric source (PS) was driven with a 22-kHz carrier signal that was up- and down-shifted by one half the difference frequency. The repetition frequency was fixed at either 1 Hz or 2 Hz and the pulse length varied from 5 to 15 msec. Calibration of the source was performed in situ. At 500 Hz, the beamwidth and source level was 8.5° and 167 dB re 1  $\mu Pa$  respectively. Although parametric

sources have the advantage of low-frequency directionality, bubbly liquids are highly nonlinear and the possibility of enhanced parametric interaction within the cloud needed to be addressed. Therefore, a conventional source (CS) (Honeywell HX-29) was also utilized. The working depth of 87.6 m was chosen to minimize reverberation.

The receivers consisted of four spherical hydrophones (ITC, 15.2 cm), situated as indicated in Fig. 4 (see H1 through H4). All hydrophone outputs were band-pass filtered, preamplified, and recorded on analog tape. Sound speeds were measured daily. At

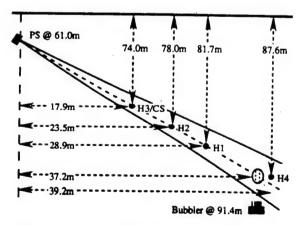


Figure 4. Test range for the backscattering experiment.

depths greater than 70 m , isovelocity conditions prevailed, with speeds of 1421.7 m/sec and 1420.8 m/sec at 70 m and 91.4 m respectively. For the calculations that follow, a speed of 1421.5 m/sec was used. We aligned the range using the 22-kHz signal to measure time-

of-flight and verify alignment and slant ranges. We then inserted a 1.12-m diameter, hollow, steel spherical target in order to test our ability to measure target strength (TS). Assuming a perfectly reflected surface, this sphere possessed a theoretical TS of -11.1 dB at 5 kHz. Our measured TS was -12.3 dB.

Bubble clouds were produced at 91.4-m depth using a pressurized steel enclosure, vented to the lake via a concentric circular array of 48 22-gauge hypodermic needles connected to a bank of 24 solenoid valves. A 3.5-sec burst of air at 68.95 kPa (10 psi) overpressure yielded a roughly cylindrically shaped cloud length  $\approx 1.7 \pm 0.3$  m, radius  $\approx 0.24 \pm 0.01$  m, with a void fraction of  $\approx 0.40 \pm 0.007$ %. The bubble sizes, which were measured with a video camera at depth as well as in a pressurized tube, were normally distributed about a peak at 1.1  $\pm$  0.2 mm radius, which corresponds to f<sub>o</sub>  $\approx$  9.2  $\pm$  1.7 kHz at a depth of 87.6 m.

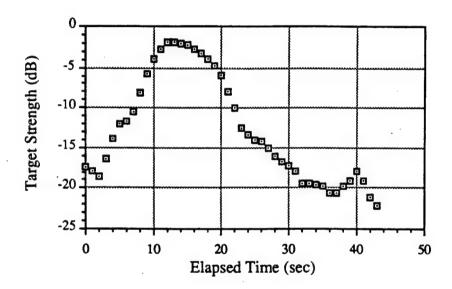


Figure 5. The evolution of the TS as the cloud traverses the beam of the parametric source operating at a pulse repetition frequency of 1 Hz

The clouds rose at ~ 0.3 m/sec, yielding a time-varying echo that reached a maximum level as the cloud crossed the range axis. Fig. 6 depicts the evolution of the backscattered target strength as the cloud rises through the beam. This was obtained by time-gating the sequence of band-pass filtered echo returns (the filter bandwidth was set to equal the signal bandwidth; i.e., 200 Hz). Rms voltages were computed for both the main bang signal (mv) and for each echo return (ev). These values where inserted, along with the transmission loss factor (TLF), into the following expression for the target strength:

$$TS = 20\log\left[\frac{ev}{mv}\right] - TLF; \quad TLF = TL_{sr} - TL_{st} - TL_{tr}, \tag{36}$$

where the TLF accounts for source-receiver, source-target, and target-receiver transmission losses, assuming spherical spreading of the propagating waves.

#### 5.2 Experimental Results.

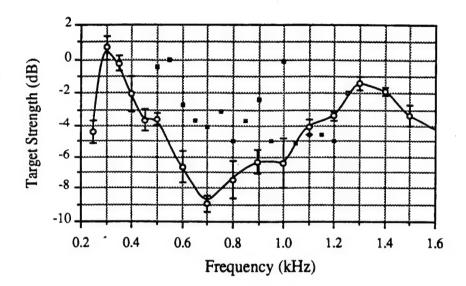


Figure 6. Backscatter TS measurements for conventional (squares) and parametric (circles) sources.

Peak target strength measurements obtained using conventional and parametric sources are plotted in Fig. 6. The target strengths are very high; at 300 Hz, the TS is about 12 dB greater than one would expect from a 1-m diameter perfectly reflecting sphere at 5 kHz. Each point corresponds to a mean value computed over a population of 4-5 clouds run at each frequency. The uncertainty corresponds to the standard error of the mean. There are apparent resonance peaks at 300 Hz and 1.3 kHz. If these peaks were due to single-bubble resonance scattering, the bubbles would possess radii of 3.4 cm and 7.8 mm respectively, which is highly unlikely in light of our revised observations of the bubbles generated at the needles and of the cloud as a whole.

It is probable that these peaks are the result of resonance scattering due to collective oscillations of the bubble cloud. The resonance frequency for a spherical bubble cloud is given by equation 35, where  $\chi$ ,  $\gamma$ ,  $P_{\text{o}}$ ,  $\rho$ , and a are, respectively, the void fraction, the polytropic exponent, the ambient pressure, the water density, and the cloud radius. We assume that the spherical cloud which has the same volume as the cylindrical cloud will describe the resonance frequency. Such a cloud, with  $\chi$  = 0.40 ± 0.07% and a = 0.42 ± 0.04 m, has an isothermal resonance frequency of 324 ± 53 Hz. Moreover, the target strength is given by:

$$TS = 10 \log \left[ \frac{I_s}{I_i} \right]_{r=1m}, \tag{37}$$

where I, and I, are the scattered and incident intensities respectively. From equations 30 and 32 this is given by

$$\frac{I_s}{I_1} \approx \frac{k^4 a^6}{9} \frac{\left[1 - \frac{y}{\overline{y}}\right]^2}{\left[\left(1 - \left(\frac{y}{\overline{y}}\right) \frac{z_o^2}{3}\right)^2 + \left(\left(\frac{y}{\overline{y}}\right) \frac{z_o^3}{3}\right)^2\right]}, \quad r = 1m,$$
(38)

where  $z_0=ka$ ,  $y=\rho c^2$ , and  $\overline{y}=\rho_a c_a^2$ ; the subscript denotes the effective properties of the air/water mixture. For the cloud in question, Equations 37 and 38 yield a calculated TS of -3.9 ± 0.8 dB at approximately 309 Hz. Although the measurements and calculations of the fundamental resonance frequency of the bubble cloud agree to within the stated uncertainty, the calculated target strengths are about 4 dB too low. All uncertainties on these results are based on either statistical analysis or ad hoc assumption and are propagated through the calculations by Taylor expanding relevant quantities to first order in the perturbed parameters. One such assumption was that the incident wave from the parametric source decreased as 1/r between the hydrophone and the target. This could lead us to underestimate the true P, and subsequently overestimate the target strength. Consequently, the difference between our measurements and calculations may be attributed to the assumption of propagating spherical wavefronts for the parametric source and/or an underestimate of the damping within the cloud.

Parametric and conventional measurements exhibit "reasonably" good agreement despite the fact that reverberation made it difficult to obtain CS data for frequencies less than 1 kHz. It appears that enhanced parametric excitation in the region occupied by the bubbly fluid did not serve to appreciably bias the TS measurements.

#### 6. SUMMARY AND CONCLUSIONS

We have reviewed the classical theory of sound radiation and scattering from compact bubble clouds. The theory shows that the important factor is the compressibility of the mixture which is characterized by the volume fraction at the lower frequencies. The volume pulsation of the cloud is the important mechanism whereby the cloud radiates or scatters sound; Strasberg has shown for the single bubble that shape may not be important. Since any compact radiator may be expanded in terms of a distribution of simple sources such as monopoles, dipoles, and quadrupoles. An expansion of the radiation field in terms of spherical harmonics identifies the monopole as the oscillation of a spherical source. We have shown that in this limit the cloud of bubbles, radiates with a resonance frequency described by a Minnaert equation modified to include the volume fraction of the gas.

Secondly, we have presented experimental results on the back-scatter from a cloud of bubbles in the absence of boundaries. The results were interesting because of the high target strengths and the low-frequency resonance behavior. The experimental and theoretical results were shown to agree in a qualitative manner, we estimated the resonance frequency correctly, but were several dB off with respect to target strength. These results should be considered preliminary as analysis of the data is continuing. Nevertheless, they clearly show the effect of low-frequency resonant scatter.

The presence of the sea surface at these low frequencies may to first order be treated by the method of images. Hence a compact

source at a depth below the pressure release surface will have a dipole characteristic and a magnitude which scales as  $(z/\lambda)^2$ , whereas a scatter below this surface will have a dipole pattern squared and a  $(z/\lambda)^4$  dependence. Therefore, the large target strengths observed here may be offset by the proximity of the rough sea surface; the deeper plumes are the most important.

#### 7. ACKNOWLEDGEMENTS

This paper represents the work of a large group of individuals. Jim Fitzgerald, The Kildare Corporation, contributed to this work through his observation of the resonant behavior of bubble clouds. Michael Nicholas of the NCPA was instrumental in all phases of our experiment. Larry Crum, NCPA, Andrea Prosperetti of The Johns Hopkins University, and Murray Korman of the U.S. Naval Academy provided critical insight, advice and criticisms. The experimental team of Jeff Schindall, Bill Konrad, Bill Marshall, Martin Wilson, Lynn Carlton, and the Lake Seneca Crew made this work possible. Ralph Goodman, The Pennsylvania State University/Applied Research Laboratory, has also contributed by his interesting discussions, technical reviews, and moderating influence. Finally, we acknowledge the sponsorship of Marshall Orr, ONR (11250A), Ken Lima, NUWC IR, and additional support from AEAS, ONT, and DARPA.

#### NOMENCLATURE

$A_{m}, \overline{A}_{m}$ $C$ $H_{m}$ (kr) $I$ $J(kr)$ $N$ $P_{\omega}, P_{i\omega}, P_{\omega}$ $P$ $P_{n}(\theta)$ $R$ $S, S_{o}, S_{a}$ $TS$	series coefficients speed of sound Hankel function intensity Bessel function number pressure transform ambient pressure source pressure Legendre polynomial radial distance surfaces target strength (dB)	i k 1 m mv fi o r s t v, v, v, v, y, y	subscript incident wave number liquid subscript mixture or index main bang voltage normal to S. radial distance subscript, scatter time normal velocities bulk modulii
TL V,V <sub>o</sub> ,V <sub>a</sub>	transmission loss (dB) volumes	Υ Υκ, Υ <sub>Ρ</sub>	gas constant compressibility,
a â $ c_m, c_{mlf}, c_g, c_e, \overline{c}_e $ ev $ f_o, f_n(\omega_o, \theta, r_n) $ $f_\omega(r)$ $g_\omega(r r_o)$ $g$	radial distance normal to S. sonic speeds  echo voltage natural frequency bubble response source function Green's function subscript for gas	δ ε θ κ λ μ π ρ χ ω Γ	density ratio dampening small quantity angular measure compressibility wave length wave number pi density void fraction angular frequency gamma function

#### REFERENCES

 Thorpe, S. A. (1982), "On the clouds of bubbles formed by breaking wind-waves in deep water and their role in air-sea gas transfer," Phil. Trans, Roy. Soc. A 304, 155-185.

- Su, M. Y., Green, A.W., and Bergin, N.T. (1984), "Experimental studies of surface wave breaking and air entrainment," in W. Brutsaert and G. Jirka (eds.), Gas Transfer at Water Surfaces, Reidel Press, pp. 211-219.
- 3. Monahan, E. C. and MacNiocaill (eds.) (1986), Oceanic Whitecaps and Their Role in Air-Sea Exchange Processes, D. Reidel, Boston.
- 4. Monahan, E. C. and Lu, M. (1990), "Acoustically relevant bubble assemblages and their dependence on meteorological parameters," IEEE Jour. Ocean Engr. 15(4), 340-349.
- 5. McDaniel, S. (1988), "High-frequency sea surface scattering: Recent Progress," J. Acoust. Soc. Am 84(S1), S121 (also: ARL/PSU T.M. 88-134-21, July 1988)
- 6. Carey, W., Goodman, R., McDaniel, S., Jackson, D., Deferrare, H., and Briscoe, M. (August 1985), "Sea surface scatter experiments, a review," Memo to M. Orr, ONR (11250A).
- 7. McDonald, B. E. (1991), "Echoes from vertically striated subresonant bubble clouds: A model for ocean surface reverberation," J. Acoust. Soc. Am 89(2), 617-622.
- Henyey, F. (1991), "Acoustic scattering from ocean microbubble plumes in the 100 Hz to 2 kHz region," J. Acoust. Soc. Am 90(1), 399-405.
- Prosperetti, A. (1988), "Bubble related ambient noise in the ocean," J. Acoust. Soc. Am. 84(3), 1042-1054.
- 10. Prosperetti, A. (1988), "Bubble dynamics in ocean ambient noise," in B. R. Kerman (ed.), Sea Surface Sound, Kulwer Acad., Boston, MA, pp. 171-171.
- 11. Carey, W. M. and Browning, D. G. (1988), "Low frequency ocean ambient noise: Measurements and theory," in B. R. Kerman (ed.), Sea Surface Sound, Kulwer Acad., Boston, MA.
- 12. Carey, W. M. and Fitzgerald, J. W. (1987), "Low frequency noise and bubble plume oscillations," j. Acoustic soc. Am. 82(51), p. 362 (paper 001) (also available NUSC TD 8495, 24 February 1989; DTIC AD206537)
- 13. Carey, W. M., Fitzgerald, J. W., and Browning, D. G. (in press 1990), "Low-frequency noise from breaking waves," in B. R. Kerman (ed.), Natural Physical Sources of Underwater Sound, Kluwer Acad., Dordrecht, Netherlands, (also: NUSC Tech. Rep. 8783, 5 oct., 1990, avail. DTIC AD227969).
- 14. Foldy, L. (1944), "Propagation of sound through a liquid containing bubbles," paper, Columbia Univ. NRDC DIV 6.1. (Available DTIC).
- 15. Spitzer, L. (1943), "Acoustic properties of gas bubbles in a liquid," paper, NRDC DIV 6.1, (available DTIC).
- 16. Foldy, L. (February 1945), "Multiple scattering of waves," Phys. Rev. 67(3&4) 107-119.

- 17. Carstensen, E. L. and Foldy, L. (May 1947), "propagation of sound through a liquid containing bubbles," J. Acous. Soc. 19(3), 481-501.
- 18. Morse, P. M. and Feshback, H. (1943), Methods of Theoretical Physics, Part II, McGraw Hill Book Co., N.Y.
- 19. Morse, P. M. and Ingard, K. (1968), Theoretical Acoustics, McGraw Hill Book Co., N.Y., pp. 413-414.
- 20. Rschevkin, S. N. (1963), The Theory of Sound, Pergamon Press Ltd., N.Y.
- 21. Anderson, V. (1950), "Sound scattering from a fluid sphere," J. Acous. Soc. Am. 22(4), 426-431.
- 22. Wood, A. B. (1932/1955), A Textbook of Sound, G. Bell and Sons Ltd., London, pp. 360-364.
- 23. Commander, K. and Prosperretti, A. (1989), "Linear pressure waves in bubbly liquids: Comparison between theory and experiments," J. Acoust. Soc. Am. 85(2), 732-746.
- 24. Gilbert, K., Wang, L., and Goodman, R. (1991), "A stochastic model for scattering from near-surface oceanic bubble layers," J. Acoust. Soc. Am. 90(s4), 2301.
- 25. Strasberg, M. (1956), "Gas Bubbles as Sources of Sound in Liquids," J. Acoust. Soc. Am. 28(1), 20-26.
- 26. Crighton, D. G. and Williams, J. E. (1969), "Sound generation by turbulent two-phase flow," J. Fluid Mech. 36, 585-603.
- 27. Van Wijngaarden, L. (1958), "On the equation of motions for mix-tures of liquid and gas bubbles," J. Fluid Mech. 133(P3), 465-474.
- 28. Karplus, H. B. (1958), "The velocity of sound in a liquid containing gas bubbles," paper, Armour Research Foundation, University of Illinois, Proj. c00-248, TID-4500.
- 29. Ruggles, A. (1987), "The Propagation of Pressure Perturbations in Bubbly Air/Water Flows," Ph.D. Thesis, Dept. of Mech. Eng., Rensselaer Polytechnic Institute, Troy, NY.

### Appendix C

### LOW FREQUENCY RESONANCE BACKSCATTER FROM NEAR-SURFACE BUBBLE CLOUDS

L. A. Crum, J. A. Schindall, and R. A. Roy
Applied Physics Laboratory
University of Washington, Seattle WA 98105 USA
and
W. M. Carey
Advanced Research Projects Agency
3701 N. Fairfax Drive
Arlington, VA 22203-1714 USA

#### Abstract

When active sonar systems are used to insonate the sea surface, anomalous scattering is observed in the form of enhanced backscatter, and more importantly, in the form of discrete, bright echoes. The most plausible explanation for these effects is the increased scattering resulting from the presence of bubble plumes and clouds, produced near the surface by breaking waves. This paper describes some preliminary calculations of the backscattered target strengths expected on the basis of resonance scattering from bubble clouds.

### l. Introduction

In tests of low frequency active sonar systems, false targets have arisen when insonation of the sea surface is attempted, especially in circumstances of high sea state [Gauss et al., 1992]. The origin of these false targets is still unclear, although the most likely candidates are assemblages of gas bubbles in an acoustically compact form. Bubble clouds are a common occurrence in the near surface of the ocean when breaking waves are present; these clouds are likely to result from bubble entrainment during wave breaking. Collections of bubbles in diffuse concentrations in the form of plumes have been observed at depths of several meters [Monahan, 1971; Thorpe, 1982; Farmer and Vagle, 1988], presumably drawn to these depths as a result of convective flows such as Langmuir circulation and thermal mixing [Thorpe, 1982]. Significant acoustic backscattering from the sea surface can result either from these small, relatively dense clouds that are near the surface, or from the larger, relatively diffuse plumes that can extend to greater depths.

It has been shown that the available surface scattering data [Chapman-Harris, 1962; Ogden and Erskine, 1992] can be accounted for in terms of either weak scattering (the well-known Born Approximation) from deep, diffuse bubble plumes generated by Langmuir circulation [MacDonald, 1991; Henyey, 1991], or by resonance scattering from higher void fraction clouds near the surface [Prosperetti and Sarkar, 1992]; it is the contention in this paper that the observed large target strength false echoes result principally from detatched bubble clouds; furthermore, we present in this paper the range of bubble cloud and environmental parameters that are likely to result in these bright targets.

### 2. Background

The attempts to characterize acoustic scattering from the ocean surface in the absence of bubble clouds, due to Bragg Scattering alone, have resulted in significant disagreement between the calculations [McDaniel, 1987] and the experimental data [Chapman and Harris, 1962; Ogden and Erskine, 1992]. Consequently, MacDonald [1991] and Henvey [1991] have used weak scattering theory (Born approximation) to obtain the surface backscatter in the presence of "tenuous" (void fractions less than, say, 10-3 %) bubble plumes of various configurations and orientations. Their calculations assume that the clouds are sufficiently diffuse so that multiple scattering can be ignored; consequently. the scattered sound energy is mostly specular. Their results indicate that if one wishes only to account for the average surface backscatter, then the tenuous bubble plumes generated by Langmuir circulation are sufficient. However, it is not yet clear whether these approaches can account for the presence of the "bright echoes" or "hot spots" observed in the critical sea tests and described by Gauss et al., [1992]. Thorsos [1992] has examined the effect of a rough surface and noted that focusing from appropriate contours can significantly enhance the calculated backscatter from tenuous clouds. However, we shall follow a different approach and assume that there are concentrations of bubbles in the form of clouds that are of sufficient void fraction to lead to a resonance oscillation of the cloud itself, thus resulting in high target strengths at low frequencies.

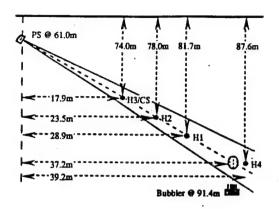
These cloud oscillations are called "collective oscillations" and represent a type of acoustic backscatter that is fundamentally different from that described by Born-approximation, weak-scattering theory. Collective effects occur when the acoustic wavelength is considerably larger than the dimensions of the cloud and the resonance frequency of the individual bubbles comprising the cloud is much higher than the insonation frequency; thus, all the bubbles oscillate essentially in phase. Because the compressibility of the cloud is similar to that for a single gas bubble while the induced mass is associated with that of the entire cloud, the oscillation frequency can be quite low, and particularly, much lower than that of an individual gas bubble. An alternative but equivalent explanation is that because the phase speed in the bubbly mixture is greatly reduced (sometimes even below that for a pure gas), the oscillation frequency is correspondingly reduced.

### 2.1 Collective Oscillations

Carey and Browning, [1988] and Prosperetti [1988] independently suggested that bubble clouds whose geometrical dimensions were small with respect to a wavelength could behave as a compact scatterers. Evidence for the existence of collective oscillations have been firmly established by laboratory work [Yoon, et al., 1991; Nicholas, et al., 1992; Lu, et al., 1991], and also by field experiments in a large fresh water lake [Roy, et al., 1992]. Furthermore, recent data by Farmer and Ding, [1992] on sources of ambient noise in the ocean provide strong support for the existence of low frequency emissions indicative of collective oscillations. If these clouds are observed to radiate at low frequencies by collective effects then it is likely that they would also act as effective scatterers of low frequency sound.

### 2.2 Preliminary Results

We have performed some preliminary experiments of the low frequency scattering characteristics of bubble clouds. The test plan and some initial results are described in a previous SACLANT Symposium report [Carey and Roy, 1993], and in a more widely distributed publication [Roy, et al., 1992]; they are shown for completeness in Fig. 1.



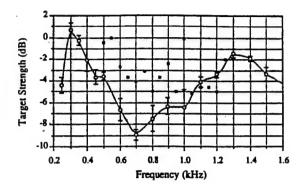


Fig. 1a. Test Plan for Seneca Lake scattering experiment.

Fig. 1b. Low-frequency backscattering results.

The principal results of this experimental study can be summarized as follows:

- Artificially generated bubble clouds of ellipsoidal geometry and about 0.5 meter in diameter and 1.0 meter in length were created at a depth of about 90 meters in a fresh water lake. The clouds were insonified with both a directed beam (parametric array) and an omnidirectional conventional source over a frequency range from 200 Hz to approximately 14 kHz.
- Measurements of the target strength (TS) as a function of frequency show relative maxima at approximately 0.3 and 1.3 kHz, as well as several other (higher) frequencies. The amplitude of these peaks is quite large and indicative of resonance effects.
- The resonance frequencies of the individual bubbles comprising the clouds are on the order of 2-3 kHz, and are so much larger than the low frequency maximum that these low frequency peaks are most likely due to collective oscillation resonances. If the cloud is treated as an acoustically compact object with a velocity of sound significantly different than that of water, then the fundamental (monopole) resonance frequency depends solely upon its volume, and its effective acoustic impedance. Using a modified Minnaert formula, a calculated resonance frequency of about 324 Hz can be obtained for the lowest peak; this value compares favorably with the measured resonance of 310 Hz [Roy, et al., 1992; Carey and Roy, 1993].
- The target strength of these clouds insonified near resonance is on the order of 0 db. Thus, they represent bright targets and compact scatterers. Calculations of the target strength based on resonance scattering [Roy, et al., 1992; Carey and Roy, 1993] suggest values on this order.

The success of these preliminary studies has emboldened us to attempt a more systematic and detailed analysis of low frequency resonance scattering from near-surface bubble clouds; our progress along these lines is described in the sections to follow.

### 3. Approach

We follow the approach of Morse and Ingard [1968] in which we assume a plane wave incident on a compliant sphere of radius  $a_c$  surrounded by a continuous medium of density and sound speed  $\rho$  and c respectively. Likewise, we consider the sphere to be a homogeneous medium of density and sound speed  $\rho$ , and c respectively. (It should be noted that the subscript " $\ell$ " refers to the bubbly mixture, not the pure liquid.) We shall assume that the target is illuminated by plane waves and that it radiates a spherically outgoing acoustic wave.

We shall also assume that the sphere (bubble cloud) is composed of many bubbles; thus, the medium is dispersive with effective density and wave number given by,

$$\rho_{\ell} = \beta \rho_{air} + (1 - \beta)\rho$$

$$k_{\ell}^{2} = k^{2} + \frac{4\pi\omega^{2}a\eta}{\omega_{o}^{2} - \omega^{2} + 2ib\omega},$$
(1)

where  $k = \omega / c$  is the wave number in the liquid, a is the radius of individual bubbles (considered to monodispersed in size),  $\eta$  is the number of bubbles per unit volume,  $\omega_o$  is the resonance frequency of the bubbles,  $\beta = 4\pi\eta a^3/3$  is the void fraction, and b is the damping constant [Commander and Prosperetti, 1989; Lu, et al, 1990]. For  $\omega << \omega_o$ , one can show that the real portion of the complex phase speed in the mixture is given by,

$$c_{\ell}^{2} = \frac{\gamma P_{\infty}}{\gamma P_{\infty}^{2} + \beta \rho}.$$
 (2)

Here,  $\gamma$  is the ratio of specific heats of the gas. It should be noted that for  $\beta$  not too small or  $P_{\infty}$  not too large, the low frequency phase speed in Eq. (2) reduces to the more familiar expression,

$$c_{\ell} = \sqrt{\frac{\gamma P_{\infty}}{\beta \rho}} \,. \tag{3}$$

For the scattering problem in an infinite medium we solve the Helmholtz equation subject to the boundary conditions of continuous pressure and normal velocity across the surface of the sphere.

$$\nabla^2 p_{\omega} + k^2 p_{\omega} = -f_{\omega}(\bar{r}). \tag{4}$$

We take the solution to be a superposition of incident plane wave and scattered waves:

$$p_{\omega} = p_i + p_s$$
 where  $p_i = Ae^{-i\vec{k}\cdot\vec{r}}$ . (5)

Morse and Ingard used an integral Green's function method to demonstrate that the solution for the exterior scattered wave in spherical coordinates is represented by an expansion in Legendre polynomials and spherical Hankel functions with appropriate coefficients,

$$p_{s}(r) = -\frac{1}{2} A \sum_{m=0}^{\infty} i^{m} (2m+1) (1+R_{m}) P_{m}(\cos \theta) h_{m}(kr)$$

$$\to i \frac{A}{2k} \frac{e^{ikr}}{r} \sum_{m=0}^{\infty} (2m+1) (1+R_{m}) P_{m}(\cos \theta)$$
(6)

where the asymptotic form is given in the far field. The coefficient  $R_m$  satisfies the boundary conditions and describes the reflectivity of the sphere where,

$$(1+R_m) = 2\frac{j_m'(ka_c) + i\beta_m j_m(ka_c)}{h_m'(ka_c) + i\beta_m h_m(ka_c)}, \text{ and } \beta_m = i\frac{\rho c}{p_\ell c_\ell} \left[ \frac{j_m'(k_\ell a_c)}{j_m(k_\ell a_c)} \right] \text{ is the specific}$$

admittance of the surface. In this study we make use of the limiting form of  $c_{\ell}$  given in Eq. (2) which is not a complex phase speed and hence does not take into account damping from the bubbles within the cloud.

We are primarily interested in the backscattering; hence, we take  $\theta = \pi$ . In the free-field, the TS is given by [Urick, 1967],

$$TS_{ff} = 20\log\left|\frac{p_s}{p_i}\right|_{r=1m} = 10\log\left(\frac{I_s}{I_i}\right)_{r=1m}$$
 (7)

Carey and Roy [1993] have shown that for small  $ka_c$ , the monopole term in Eq. (6) can be approximated by

$$\frac{p_s}{p_i} \approx i \frac{k^2 a^3}{3r} \frac{\left(1 - \frac{\rho c^2}{\rho_t c_t^2}\right)}{\left(1 - \frac{\rho c^2}{\rho_t c_t^2} \frac{(ka_c)^2}{3}\right) - i \left(\frac{\rho c^2}{\rho_t c_t^2} \frac{(ka_c)^3}{3}\right)}.$$
 (8)

At resonance, this leads to

$$\omega_c \approx \frac{1}{a_c} \sqrt{\frac{3\gamma P_{\infty}}{\beta \rho}},\tag{9}$$

which is a modified form of Minnaert's equation for bubble resonance.

The above equations allow us to make some observations concerning the interdependence of some of the more important parameters, viz,

- The effective phase speed in the bubbly mixture increases with depth, for a fixed volume fraction  $\beta$ ; likewise, for fixed  $a_c$ , the resonance frequency of the cloud,  $\omega_c$ , also increases with depth.
- As  $\beta$  decreases/increases, both  $\omega_c$  and  $c_\ell$  increases/decreases (other parameters held fixed).

• As  $a_c$  decreases/increases,  $\omega_c$  increases/decreases (other parameters held fixed).

Furthermore, we can also deduce how changes in some of these parameters affect the TS? At resonance, Eq(8) reduces to

$$\left| \frac{p_s}{p_i} \right|_{r=1_{m}} \to \frac{c}{\omega_c} \left( \frac{\rho_t c_t^2}{\rho c^2} - 1 \right). \tag{10}$$

Thus, as the resonance frequency decreases, or the yold fraction becomes larger, or the size of the cloud increases, the TS increases.

We now turn to a determination of the TS for a variety of environmental and acoustic conditions. In our calculations, we choose to obtain the resonance frequencies by numerical methods. We will approximate Eq. (6) by truncating the series at the m=0 monopole term, since for the frequencies of interest in this paper, the wavelength of sound is many times greater than the bubble cloud radius and the higher order terms do not contribute to the lowest resonance.

To solve the problem of scattering from targets near the ocean surface we make use of the method of images; in Fig 2 below a diagram is shown of our approach.

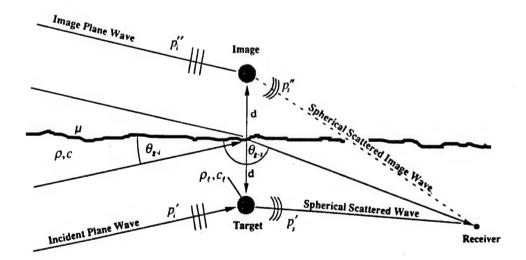


Fig. 2. Diagram of the theoretical model used in the determination of the scattered target strengths.

Here we treat the target as a point scatterer a distance d below a pressure release surface with reflectivity coefficient  $\mu$  (-1<  $\mu$ <0). The reflection coefficient describes the roughness of the sea surface where  $\mu$  = -1 corresponds to a smooth pressure release surface and  $\mu$  = 0 corresponds to an extremely rough surface (effectively equivalent to an infinite medium); realistic sea states fall somewhere in between. Rather than relate  $\mu$  to the frequency and wave-height, we have chosen to evaluate the TS at fixed values of  $\mu$  in order to generalize the analysis. Using the method of images one obtains,

$$p_i = p_i' + p_i'' = p_i \left[ e^{i(kx\cos\theta_s + kd\sin\theta_s)} + \mu e^{i(kx\cos\theta_s - kd\sin\theta_s)} \right]$$
 (11a)

$$p_s = p_s' + p_s'' = \frac{p_s}{r} \left[ e^{i(kx\cos\theta_s + kd\sin\theta_s)} + \mu e^{i(kx\cos\theta_s - kd\sin\theta_s)} \right], \tag{11b}$$

where  $\theta_g$  is the surface grazing angle, and  $p_i$  and  $p_s$  are the magnitudes of the incident and scattered fields in the free-field. The single-primes indicate the fields neglecting the surface, the double-primes denote the image fields, and  $|p_s|^2 = |P_i|^2 f^2$  is the response of the cloud to the incident plane wave, where  $TS_f = 10\log f^2$  is the target strength computed in a free field. After some algebra, we find that the TS for a bubble cloud near the sea surface is given by

$$TS = 20 \log \left| \frac{p_s}{p_i} \right|_{r=1m} = TS_{ff} + 10 \log \left\{ 1 + \mu^2 + 2\mu - 4\mu \sin^2(kd \sin \theta_g) \right\}^2.$$
 (12)

Clearly, when  $\mu = 0$  the expression for TS yields the free-field target strength; for  $\mu = -1$ , and the source close to the surface, phase cancellation occurs, and the source behaves like a dipole.

If one considers the limiting from of Eq. (12), for kd<<1, one observes a dipole characteristic to the scattered field in which the scattered pressure scales with  $(d/\lambda)^4 \sin^4\theta$ , where  $\lambda$  is the acoustic wavelength. This equation suggests a complex acoustical behavior which can be briefly summarized as follows. High void-fraction clouds which generate significant TS's in the free field may not be acoustically important because these clouds tend to reside near the surface, and thus are subject to the mitigating effect of surface dipole cancellation. This effect is exacerbated by the fact that these large TS clouds tend to resonate at low frequencies (i.e. long wavelengths), and the proximity of the cloud to the surface is defined relative to the acoustic wavelength.

If one considers deeper scatterers, then one is necessarily limited (by oceanographic constraints) to the consideration of lower void-fraction clouds that will not have as pronounced a resonance scattering characteristic. Indeed, clouds in the deepest portion of the bubble layer, (order 10 m) are very tenuous and probably do not resonate at all. It seems likely that there are optimum combinations of cloud depth, cloud characteristics and frequency that produces significant backscatter target strengths (larger than, say -10 db). In the next section, using the equations determined above, we have explored a variety of conditions that could give rise to significant scattering TS's.

### 4. Results

We have generated a series of multidimensional figures to display the calculated target strengths of these bubble assemblages as a function of several relevant parameters. These figures are quite complex and require studious attention in order to fully grasp the principal implications of the data. The parameters shown in these figures with brief comments where appropriate are as follows: TS--the target strength, defined as in Eq. 12 above;  $\beta$ -the void fraction, defined above; d--the depth of the center of the bubble cloud below the ocean surface;  $\theta_g$ --the grazing angle of the incident sound beam;  $a_c$ --bubble cloud radius;  $\mu$ --a measure of the reflectivity of the surface: for  $\mu$  = -1, the surface is perfectly reflecting, for  $\mu$  = 0, the surface is at infinity and there is no reflection from the surface.

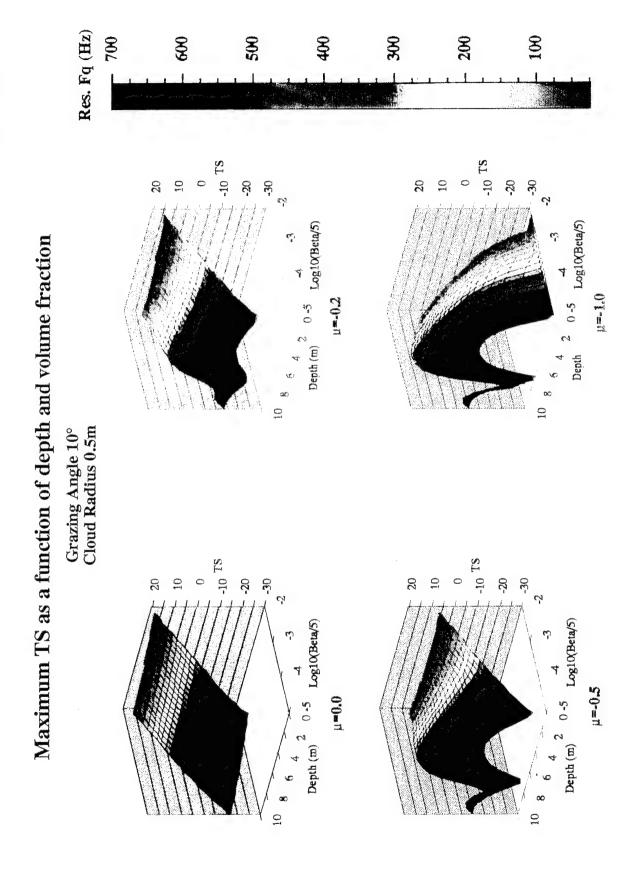
Consider Fig. 3, which is a four-dimensional plot of the target strength as a function of void fraction, cloud depth, and cloud resonance frequency. For this case, the grazing angle is shallow-- $10^{\circ}$ --and the cloud radius is relatively large--0.5 m. We anticipate that this case would correspond to the insonation of a "bubble plume", formed from the convection of entrained gas bubbles to a considerable depth; however, this plume will be treated as a resonant, compact scatterer. We presume that the large bubbles have risen by gravitational forces to the surface, and that consequently the remaining bubbles, and particularly the void fraction, are both relatively small. The data shown in the lower right-hand-corner of Fig. 3 correspond to  $\mu$  = -1, which presumes a perfectly reflecting surface. Note that there is little scattering for small depths; in this case the cloud acts as a dipole scatterer, and thus for small grazing angles, the TS is very low. However, as the depth increases to a few meters, even for void fractions as low as  $10^{-5}$ , relatively large TS's are observed. For example, at a depth of 5 m, and for a void fraction of  $10^{-4}$ , the observed target strength is on the order of 0 db for an insonation frequency of 300 Hz.

With a value of  $\mu=-1$ , it is assumed that the surface is perfectly reflecting, a situation unlikely to be realized in a rough sea. The data shown in the upper left hand corner of this figure corresponds to value of  $\mu=0$ , or for a surface so rough that no coherent energy is reflected. Note that in this case, there is little depth dependence--no phase cancellation from the reflecting surface--and this cloud is a high TS scatterer for all "depths". Perhaps a more reasonable representation of the surface effect is presented in the lower left-hand-corner, calculated for a value of  $\mu=-0.5$ , which thus corresponds to the intermediate case. It is seen from this figure, that TS's on the order of 0 db should be observed for plumes with void fractions above  $10^{-4}$ , at depths between 2-6 m, and at insonation frequencies on the order of 300 Hz. As the void fraction falls below  $10^{-5}$ , the backscattered intensity rapidly falls in magnitude.

Let us now compare and contrast these results for a relatively large, low void-fraction plume with the case shown in the next figure. Shown in Fig. 4 are plots for a bubble cloud of radius 0.1 m, and a grazing angle of  $10^{\circ}$  (as in Fig. 3). Consider again first the case for a perfect reflecting surface, as shown the lower right-hand-corner. This cloud is relatively small; thus, the monopole resonance frequency is rather high (of order 400 Hz near the surface). Consequently, it can produce a large backscattered TS even when it is within a few meters of the surface. Since we can presume that for such a small cloud, it is not unreasonable to expect a large void fraction, we shall consider values of  $\beta$  as high as  $10^{-3}$ . Note that for the surface conditions of  $\mu = -1$ , TS's of 0 db can be expected only for clouds 3-4 meters below the surface.

Consider next, however, the conditions demonstrated in the upper left-hand-corner, in which the surface condition corresponds to a very rough surface--a more likely occurrence for large void fraction clouds. For this case, TS's of 0 db can be observed at any depth for a range of frequencies near 400 Hz. Again, if the more realistic case of  $\mu$  = -0.5 is considered (lower left), a TS of 0 db can be expected to occur for a cloud with a void fraction of  $10^{-3}$ , a radius of 0.1 m, a frequency of 400 Hz, and at a depth of about 4 m.

Finally, let us consider Fig. 5, which summarizes the principal features of this report. Our field measurements at Lake Seneca [Roy, et al., 1992; Carey and Roy, 1993] demonstrated that TS's of approximately 0 db could be achieved for resonance oscillations of a bubble cloud, far removed from the surface, with a void fraction of about 10<sup>-3</sup>, a radius of about 0.25 m and at an insonation frequency of about 300 Hz. Furthermore, Prosperetti and his students [Prosperetti, et al., 1993; Sarkar and Prosperetti, 1993] have demonstrated that bubble clouds of similar size and void fraction, when located very near the surface, could account for the Chapman-Harris, Ogden-Erskine surface acoustic backscatter when



strengths are thus obtained for a range of frequencies. The effect of the surface is described via a reflection coefficient, µ. When µ is -1, the surface is Fig. 3. Three dimensional plot of the calculated maximum target strength as a function of depth and volume fraction for a grazing angle of 100 and a cloud radius of 0.5 m. The frequency of the monopole resonance is shown via the color bar to the right; this figure demonstrates that these target perfectly reflecting; when µ is 0, no reflection occurs.

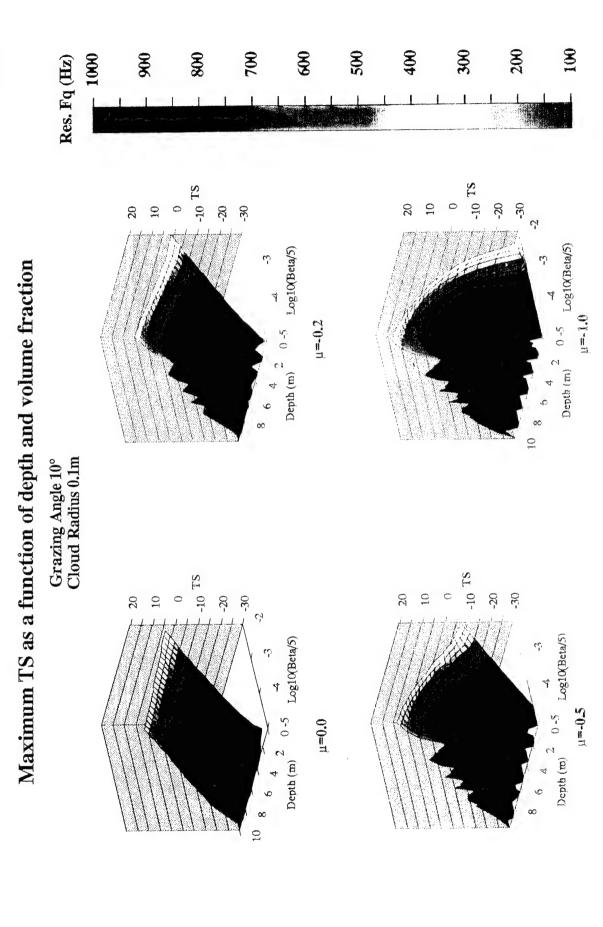


Fig. 4. Three dimensional plot of the calculated maximum target strength as a function of depth and volume fraction for a grazing angle of 10°0 and a cloud radius of 0.1 m. The frequency of the monopole resonance is shown via the color bar to the right. The effect of the surface is described via a reflection coefficient, µ. When µ is -1, the surface is perfectly reflecting; when µ is 0, no reflection occurs.

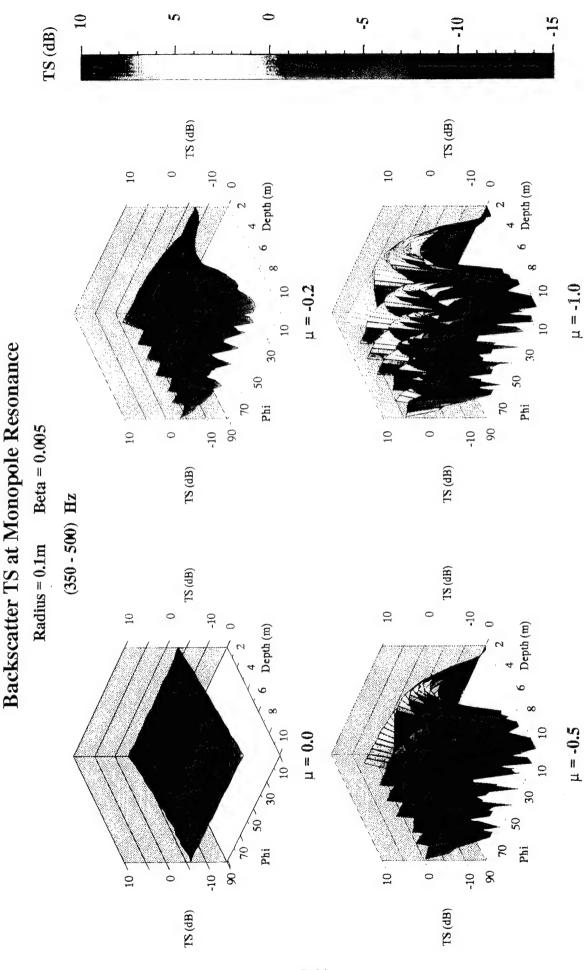


Fig. 5. Three dimensional plots showing calculations of the backscatter target strength at monopole resonance for a bubble cloud with a radius of 0.1 m and a void fraction of 0.5 %, as a function of cloud depth and grazing angle (phi). The range of resonance frequencies for this figure are from 350-500 Hz. The four different plots show the effect of the surface through the reflection coefficient,  $\mu$ .

treated as resonant scatterers. We show now in Fig. 5 the expected individual resonance scattering characteristics of these clouds as a function of such parameters as grazing angle, depth below the surface, and surface conditions.

Shown in the lower right-hand corner of Fig. 5 is the predicted backscattered target strength for a bubble cloud at its monopole resonance (varying between 350-500 Hz), for a radius of 0.1 m, a void fraction of 0.5 %, and at a perfectly reflecting surface. Also shown in this figure are the cases for a free field (upper left) and for the intermediate case of a partially reflecting surface (lower left). Consider the plot in the lower left. This case represents our estimate of the conditions expected to give rise to false echoes in low-frequency, near-surface scattering. Note that one should expect echoes with TS's of 0 db for a wide range of grazing angles and cloud depths. Bright echoes (of order 0 db) can be expected to occur for grazing angles from 10-70°, and for cloud depths from near the surface to 10 meters. This figure indicates that the ocean surface is very rich with possibilities for bright echoes from resonant bubble clouds.

The conditions that lead to the production of bubble clouds (high winds and high sea states) also favor the generation of high scattering TS's. First of all, high sea state means that the ocean surface will be rough; thus,  $\mu$ =-0.5 is not an unrealistic approximation; surface cancellation won't be as important except for very low frequencies. Secondly, stormy conditions usually imply an unstable water column, which is often upward refracting. Thus, the incident angles become steeper and the  $\sin^4\theta$  dependence is no longer as important. These two effects tend to make far-field-like behavior much more pronounced.

Finally, we make note of the experiments of Lamarre and Melville [1992] who obtained measurements of the void fractions of ocean-generated bubble clouds in the open ocean off the coast of Delaware during the last week of February, 1991. Their data indicate that significant numbers of clouds are produced with void fractions on the order of 0.5 %; they remark that their data are consistent with their laboratory results and are several orders of magnitude higher than the often-reported time-averaged values [Farmer and Vagle, 1988].

Further experiments are required, however, to determine the precise acoustic and environmental conditions that give rise to these false echoes.

### 5.0 Summary

False targets can occur for low frequency sound scattering from the sea surface. It is unlikely that Bragg scattering from a rough surface is the source of these echoes; rather, bubble clouds or plumes resulting from breaking waves represent a more plausible explanation. We have shown that bubble clouds or plumes can act as resonant scatterers at low frequencies and generate target strengths as large as 0 db for a wide variety of conditions that are expected to occur in rough seas with wind-driven breaking waves. Future experiments are planned to determine more precisely the measured targets strengths in terms of the environmental conditions that would generate bubbles clouds of the required size, shape and void fraction to produce these false targets.

### 6.0 List of References

Carey, W. M. and Browning, D., "Low-frequency ocean ambient noise: Measurement and theory," in Sea Surface Sound, B. R. Kerman Ed. (Kluwer, Boston) 361-376 (1988).

Carey, W. M., Fitzgerald, J. W., and Browning, D.G., "Low frequency noise from breaking waves," in *Natural Physical Sources of Underwater Sound*, B. R. Kerman Ed. (Reidel, Dordrecht), to appear (1993).

Carey, W. M. and Roy, R. A., "Sound scattering from microbubble distributions near the sea surface," in *Proceedings of the SACLANT Reverberation Symposium* (Kluwer, Boston), to appear (1993).

Chapman, R. P. and Harris, J., "Surface backscattering strengths measured with explosive sound sources, "J. Acoust. Soc. Am. 34, 1592-1597 (1962).

Commander, K. W. and Prosperetti, A. "Linear pressure waves in bubbly liquids: Comparison between theory and experiment," J. Acoust. Soc. Am. 85, 732 (1989).

Farmer, D. M. and Ding, L., "Coherent acoustic radiation from breaking waves", J. Acoust. Soc. Am. 92, 397 (1992).

Farmer, D. and Vagle, S., "On the determination of breaking wave distributions using ambient sound," J. Geophys. Res. 93, 3591 (1988).

Gauss, R. C., Fialkiowski, J. M., and Soukup, R. J., "Low-frequency direct-path surface and volume scattering measured using narrowband and broadband pulses," SACLANT Ocean Reverberation Symposium Proceedings (to be published).

Henyey, F. S., "Acoustic scattering from oceanic microbubble plumes in the 100 Hz to 2 kHz region," J. Acoust. Soc. Am. 90, 399-405 (1991).

Lamarre, E., and Melville, K., "Instrumentation for the measurement of void-fractions in breaking waves: Laboratory and field results", *IEEE J. Ocean Engr.* 17, 204-215 (1992).

Lu, N.Q., Prosperetti, A. and Yoon, S.W., "Underwater noise emissions from bubble clouds," *IEEE J. Ocean Eng.* 15, 275-281 (1990).

McDaniel, S. T., "Vertical spatial coherence of backscatter from bubbles," *IEEE J. Ocean. Eng.* **OE-12**, 349-356 (1987).

McDonald, B. E., "Echoes from vertically striated subresonant bubble clouds: A model for ocean surface reverberation," J. Acoust. Soc. Am. 89, 617-622. (1991).

Monahan, E. C., "Oceanic whitecaps," J. Phys. Oceanogr. 1, 139-144 (1971).

Morse, P. M. and Ingard, K., Theoretical Acoustics, McGraw Hill Book Co., N. Y., 413-414 (1968).

Nicholas, M., Roy, R. A., Crum, L. A., Oguz, H. N. and Prosperetti, A., "Sound emissions by a laboratory bubble cloud," *J. Acoust. Soc. Am.*, submitted (1992).

Ogden, P.M. and Erskine, P.T., "An empirical prediction algorithm for low-frequency acoustic surface scattering strengths," Report NRL/FR/5160-92-9377, Naval Research Laboratory, Washington D.C. 20375-5000 (1992).

Prosperetti, A., "Bubble-related ambient noise in the ocean," J. Acoust. Soc. Am. 84, 1042-1054 (1988).

Prosperetti, A., Lu, N.Q., and Kim, H. S., "Active and passive acoustic behavior of bubble clouds at the ocean surface," *J. Acoust. Soc. Am.* (in press) (1993).

Prosperetti, A. and Sarkar, K., "Enhanced backscattering from bubble cloud distributions on the ocean surface," J. Acoust. Soc. Am. 91, 2315 (1992).

Roy, R.A., Carey, W., Nicholas, M., Schindall, J. and Crum, L.A., "Low-frequency scattering from submerged bubble clouds," *J. Acoust. Soc. Am.* 92, 2993-2996 (1992).

Sarkar, K. and Prosperetti, A., "Back-scattering of underwater noise by bubble clouds," *J. Acoust. Soc. Am.*, in press (1993).

Schindall, J., Carey, W. M., Roy, R. A., Crum, L. A., and Nicholas, M., "Sound scattering from near-surface bubble clouds", J. Acoust. Soc. Am., 92, 2392 (1992).

Thorpe, S. A., "On the clouds of bubbles formed by breaking wind-waves in deep water, and their role in air-sea gas transfer," *Phil. Trans. Roy. Soc.* London A304, 155-210 (1982).

Thorsos, E. I., "The effect of the rough air-sea interface on scattering from tenuous bubble clouds," J. Acoust. Soc. Am., 91, 2317 (1992).

Urick, R. J., <u>Principles of Underwater Sound for Engineers</u> (McGraw-Hill, New York), Chap. 8, p. 188 (1967).

Yoon, S. W., Crum, L. A., Prosperetti, A. and Lu, N. Q., "An investigation of the collective oscillations of a bubble cloud," *J. Acoust. Soc. Am.* 89, 700-706 (1991).

### 7.0 Acknowledgment

We wish to acknowledge helpful discussions with A. Prosperetti and the financial support of the Office of Naval Research (Ocean Acoustics) and ARPA.

### Appendix D

In <u>Proceedings of the 2nd European Conference on Underwater Acoustics</u>, ed. by L. Bjorno, Pub. by European Commission, Luxembourg, 207-212 (1994).

The low-frequency radiation and scattering of sound from bubbly mixtures near the sea surface

William M. Careya and Ronald A. Royb

<sup>a</sup>Advanced Research Projects Agency, 3701 North Fairfax Drive, Arlington, Va. 22203, USA <sup>b</sup>Applied Physics Lab, Univ. of Washington, 1013 NE 40th St., Seattle, WA 98105, USA

The radiation and scattering from micro-bubble clouds with volume fractions greater than .0001 are described with a theoretical model and measured data. In the low-frequency limit the radiation and scattering of sound was determined to be monopole when in the free field. The natural frequencies of the clouds is shown to be predicted by a modified Minnaert equation. The measured backscatter target strength was found to be consistent with theory.

### 1. INTRODUCTION

Micro-bubble plumes are produced when waves break and are convected to depth. The fundamental question is: What role, if any, is played by these micro-bubble plumes in the production of sound and the scattering of sound near the sea surface from the low (20 Hz) to mid (2kHz) frequency range? Measurements of ocean ambient noise show a dramatic increase in mid-frequency ambient noise levels when wave breaking occurs. Furthermore, measurements of the sound scattering from the sea surface has a different characteristic than expected by Bragg scattering from gravity waves, i.e. a large zero Doppler component. We hypothesized that if micro-bubble clouds and plumes with void fractions greater than .0001 act as collective resonant oscillators, then radiated noise can be produced and scattering can occur with little Doppler shift but ample spread. This hypothesis was based on the theory that when a large number of micro-bubbles with individual resonant frequencies far above the frequency of excitation are present in an acoustically compact region, the mixture properties determine the radiation and scattering. Our experiments have shown that the radiation of sound from a compact region can be describe by monopole volume pulsation beneath a pressure release surface, a doublet with a dipole radiation characteristic and natural frequency described by the volume-fraction-modified Minneart formula. Likewise, our low-frequency scattering measurements from submerged bubble clouds were consistent with the calculated resonance frequencies and target strengths.

### 2. THE OSCILLATION OF A BUBBLY SPHERICAL VOLUME

In a bubbly mixture for which the distribution of bubble sizes results in bubble resonant frequencies much greater than the frequencies of interest, the propagation of sound in the mixture can be described by an effective wave equation with mixture density and sound speed. These effective wave equations have been studied by several investigators {i.e., Foldy (1945), Commander & Prosperetti (1989), Crighton (1969), Van Wijngaarden (1968); for reviews and references see [1,2,3]}. A.B. Wood (1932) showed that, at low frequencies in an air-water mixture for which the size of the bubbles and spacing between bubbles were small compared to a wavelength, the equilibrium mixture density  $(\rho_m)$  and compressibility  $(k_m)$  are given by:

 $\rho_{\rm m} = (1-\chi)\rho_{\ell} + \chi \rho_{\rm g}; \quad \kappa_{\rm m} = (1-\chi)\kappa_{\ell} + \chi \kappa_{\rm g}; \quad \text{and} \quad C_{\rm mlf}^2 = \frac{\rm dP}{\rm d\rho} = (\rho_{\rm m}\kappa_{\rm m})^{-1} \tag{1}$ 

where  $(\chi)$  represents the volume fraction,  $\ell$  the liquid, g the gaseous component, and  $C_{mlf}$  is the low-frequency limiting sonic speed. In the case of an air-water mixture at low-frequencies, the process is isothermal, the controlling factor being the amount of heat transfer to the surrounding fluid in 1/2 of an acoustic cycle. The following expression results:

$$C_{\text{mlf}}^2 \approx P/\rho_{\ell} \chi (1-\chi), \quad \text{when } 0.001 < \chi < 0.94.$$
 (2)

This equation shows the large effect of void fraction on sonic speed. Karplus (see [3]) has verified these results in the low-frequency range (f<1kHz). Consequently at low-frequencies

only the volume fraction is important and not the bubble-size distribution.

The theory describing the generation and scattering of sound from a compliant sphere immersed in a fluid can be found in classical texts on the theory of acoustics {see references to Morse (1948), Anderson (1950), Rschevkin (1963), and D'Agostino & Brennen (1983, 1989) in [1,2]}. Here, the bubbly mixture has no well-defined boundary; but, none-the-less, is localized by perhaps a vortex or some other circulatory feature beneath the breaking wave. Following Rschevkin and Morse, we assume that the bubbly region is compact with an arbitrary radius  $r_0$  and the region is composed of micro-bubbles with resonant frequencies far above the frequency of excitation. Buoyancy forces and restoring forces such as surface tension are not required. The properties of the bubbly region are described by the mixture speed  $\overline{C}$  and density  $\overline{\rho}$ . The sound source is assumed global compared to the dimensions of the compact cloud; that is, a plane wave expanded in terms of spherical harmonics:

$$P_{i} = P_{o} \exp(i\omega t - ikr) = P_{o} \exp(i\omega t) \sum_{m=0}^{\infty} i^{m} (2m+1) P_{m}(\theta) J_{m}(kr) . \tag{3}$$

We require continuity of velocity,  $V_i + V_s = \overline{V}_n$ , and pressure,  $P_i + P_s = \overline{P}$ , at the generalized radius,  $r_0$ . Furthermore, a radiation condition is imposed at large r and the field is required to remain finite within the bubbly region. The procedure is to assume the scattered wave or radiated wave is a sum of outward propagating spherical waves:

$$P_{s} = \exp(iwt) \sum_{m=0}^{\infty} A_{m} P_{m}(\theta) H_{m}(kr) \exp t(-i\varepsilon_{m}(kr)) , \qquad (4)$$

where the term  $H_m(kr) \exp(-i\varepsilon_m(kr))$  is the spherical Hankel function of the second kind. The pressure field inside the volume  $(\overline{P})$  is expanded in terms of spherical Bessel functions of the 1st kind:

$$\overline{P} = \exp(i\omega t) \sum_{m=0}^{\infty} \overline{A_m} P_m(\theta) J_m(\overline{k}r) . \tag{5}$$

When the region is compact  $\lambda > 2\pi \, r_o$ , we can solve for the coefficients  $A_m$  and  $\overline{A}_m$  using the above boundary conditions. Equating m order terms to  $0(k^3r^3)$ , we find

$$A_{O} = \frac{iP_{O}(k_{O}r_{O})^{3} (1-y)/3}{1-y(kr_{O})^{2}/3 - iy(kr_{O})^{3}/3} ; \qquad A_{1} = \frac{P_{O}(kr_{O})^{3}(\bar{\rho}-\rho)}{2\bar{\rho}-\rho} \text{ where } : y = \rho C^{2}/\bar{\rho}\bar{C}^{2} . \tag{6}$$

When the real part of the denominator of Ao is equal to zero, a resonance has occurred:

$$(kr_o)^2 = \left(\frac{2\pi f_o r_o}{C}\right)^2 = \frac{3\delta C^2}{\rho C^2}, \quad f_o = \frac{C}{2\pi r_o} \left(\frac{3\delta C^2}{\rho C^2}\right)^{1/2} \quad \text{since}$$
 (7)

$$\overline{C}^2 = \frac{\gamma P}{\rho_l \chi (1 - \chi)}, f_o = \frac{1}{2\pi r_o} \left( \frac{3\gamma P}{\rho_l \chi (1 - \chi)} \right)^{1/2} - \frac{1}{2\pi r_o} \left( \frac{3\gamma P}{\rho_l \chi} \right)^{1/2}. \tag{8}$$

We recognize this as a modified Minnaert formula for the resonant frequency of a volume oscillation of a gas bubble. This result follows because the resonant angular frequency is proportional to the compressibility of the bubbly region, characterized by the stiffness  $(4\pi r_0 \bar{\rho} \bar{C}^2)$  and the inertia  $(4\pi r_0^3 \rho/3)$ . For air,  $C_a \approx 340 \text{m/s}$  and  $\rho_g/\rho_\ell = 0.0013$ ; we find the

resonant frequency of the gas bubble to be  $f_{ob} = 3.379/a$  (Hz). Conversely, a cloud with a v.f. of  $\chi = 0.002$  ( $C_m = 200 \text{m/sec}$ ), possesses a resonant frequency  $f_{oc} \approx 55/r_o$  (Hz). If the bubble size is 50µm, then we have  $f_{ob} \approx 67.6 \text{kHz}$ , and if the radius of the air cloud is  $r_o \approx 0.25 \text{m}$ , we find that  $f_{oc} \approx 220 \text{Hz}$ . Thus, the cloud radiates sound at a resonant frequency much lower than its constituents. Estimates for a cylindrical volume can be scaled by a comparison of the resonant frequency of a sphere,  $f_{os}$ , with a given radius at a specific void fraction. The resonant frequency of a compact cylinder,  $f_{oc}$ , can be estimated by determining it's bulk compressibility (B) and added mass (M). For a volume fraction of greater than 0.01 the equations [4] for near surface clouds are:

$$f_{os} = 2.76 \chi^{-1/2} / r_{os}; \qquad f_{oc} = (.92 \text{ to } 1.10) \chi^{-1/2} / r_{oc}$$
 (9)

These equations show that the frequency radius dependence of a cylinder with a  $\chi = 0.001$  approximately corresponds to a spherical cloud of  $\chi = 0.01$ .

### 3. THE TIPPING TROUGH EXPERIMENT [4]

The tipping trough simulated a breaking wave event under controlled mechanical and hydrodynamic conditions and was used to produce bubble clouds for the acoustic tests conducted at Seneca Lake, a water depth of 122m and at Dodge Pond, a water depth of 12.2m. The trough consisted of a 45° plexiglass wedge with a radius of 0.23m, a length of 0.61m and a volume of 7.57 liters and was positioned at a height of 0.305m above the water surface. This trough emptied into the rectangular (0.94x1.14m) opening of the salt-water containment bag (1.59mm thick butyl rubber) which was positioned just below the water surface and extended to a depth of 2.44m. The acoustic transparency and response of this bag was determined with a sound source positioned beneath the water surface. Tips were conducted with fresh water and the radiated sound pressure level spectra were measured with and without the bag and produced nearly identical results. Measurements with calibrated sound sources showed that the bottom did not provide a strong acoustic reflection.

A video microscope system was used in the laboratory to determine the bubble-size distribution and evolution of the bubble plume generated as a result of the "tipping trough". The mechanical "tipping trough" action was kept constant, while the salinity of the water was varied from  $0^{\circ}/_{00}$  (fresh water) to  $20^{\circ}/_{00}$  (salt water) [4]. The temporal evolution of the population of bubbles with radii between 180µm and 5mm was measured and showed a marked increase in the number of small bubbles as the salinity of the water increased. In this experiment the sizes of fresh and salt water bubbles had corresponding resonant frequencies greater than 0.66kHz. The fresh water bubble-size curve peaked near a frequency between 1.08 and 1.63kHz, while the salt water curves peaked near 10.87kHz. The peak temporal volume fractions for these submerged plume of both fresh and salt water were between 0.16 and 0.20. Mean volume fractions for the plumes based on the measured bubble size distributions were between 0.01 and 0.05.

Each trough tip produced transient noise characterized by the initial water entry and the subsequent evolution of the air-water mixture. The transient analysis was accomplished by selecting a specific time window referenced to the initial water entry so that the same relative time window for each tip was captured, its transient spectrum determined, and a linear average of the spectra from eight tips performed. Shown in Fig. 1a are the linear average spectra for fresh water (FW) and salt water (SW) tips in the 5-kHz analysis band with a resolution bandwidth of 12.5Hz. These spectral results shows a prominent low-frequency spectral peak near 100Hz for the FW and SW tips with the SW spectrum level (SWSL) less than the FW spectrum level (FWSL) by 3dB. The major feature observed in Fig. 1a is the peak near 100 Hz, which occurs in both SW and FW cases.

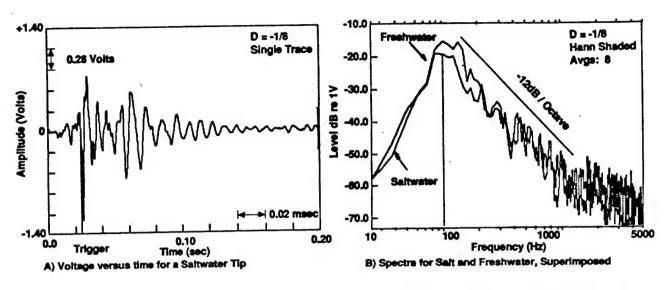


Figure 1. a) Averaged transient capture spectrum in a 5-kHz band for eight FW/SW tips. b) Single transient capture for a SW tip.

The presence of this spectral peak is significant since these cases have radically different bubble-size distributions with comparable mean volume fractions (0.01-0.05); neither distribution contains bubbles on the order of 3.3 to 3.4cm in radius which would be required to produce sound at 100Hz. Figure 1b shows a representative single event with such a pronounced characteristic. The initial water entry is indicated by the first large negative voltage peak. This is followed (~32msec) by a large damped oscillation of 89Hz which decreases within 3 cycles, followed by an increase and subsequent 6-cycle decay at a frequency of ~118Hz. This trace is remarkable insofar as it shows an example of the singular nature of the sounds produced by the air-water mixture in this tipping event. Although not all events produced clearly dampened singular sinusoids, a large number did and the remainder produced a mixture of dampened sinusoids with low-frequency content. The sequence of acoustic events in these trough tips was observed to be repeatable. These measured characteristics of the tipping trough experiments are summarized in terms of the initial potential energy of 123-169 J, the energy flux of low-frequency sound of 0.4-2.5x10-6J/m<sup>2</sup>, the energy flux source levels (EFSL) of 95-104 dB re  $\mu Pa^2$ -sec @ 1m, and the radiation efficiency between 0.3-2.26x10<sup>-8</sup>.

These results show a distinct difference between the EFSL for the case of fresh water compared to salt water; the fresh water case produces a higher spectral level. This result is very important since collective or continuum effects are thought to be isothermal at these lower frequencies. The forced oscillation of small bubbles at frequencies much less than their resonance frequencies results in the thermal dissipation of energy. Thus, as one holds the volume fraction constant and increases the number of small bubbles one would expect thermal dissipation to increase with surface area. This result implies that under conditions of constant volume fraction, the ratio of thermal loss in SW to FW clouds is proportional the ratio of

product of the number of bubbles and the mean size [4].

The trough volume corresponds to a cylindrical volume of the entrained mixture with a radius between 7.67 and 10.21cm. For a volume fraction of 0.01, the resonant frequency calculated using Eq. 9 would be between 90 and 120Hz; the range of resonant frequencies observed. We conclude that the entrained bubbly mixture oscillated in its volume mode described by the modified Minnaert's formula. That is to say a complex mixture of a bubbly liquid can be described by a monopole term -- the volume oscillation of the mixture. Due to the proximity of the pressure release surface, this monopole source of radiation and it's image behave as a dipole and only the volume mode is important in the far field.

### 5. THE MEASUREMENT OF BACKSCATTER FROM A SUBMERGED BUBBLE CLOUD [5]

The Seneca Lake low-frequency bubble cloud scattering test [5] used a moored barge in 130m of water. Equipment was deployed with the use of davits, a cable meter, and the edge of the barge, resulting in a vertical geometry. The parametric sound source, conventional transmitters and hydrophone receivers were oriented co-linearly, with the axis of the range intersecting the path of a rising bubble cloud. Though the test consisted of measurements for frequencies ranging from 250Hz to 14kHz, only the low-frequency (<1.6 kHz) results will be discussed. The parametric source (PS) was driven with a 22-kHz carrier signal that was up-and down-shifted by one half the difference frequency. The repetition frequency was fixed at either 1 or 2Hz and the pulse length varied from 5 to 15msec. Calibration of the source was performed in situ.. The beamwidth and source level at 500Hz was 8.5° and 167dB re 1µPa. Although parametric sources have the advantage of low-frequency directionality, bubbly liquids are highly nonlinear and the possibility of enhanced parametric interaction within the cloud needed to be addressed. Therefore, a conventional source was also utilized.

The outputs from four spherical hydrophones (ITC, 15.2cm) were band-pass filtered, preamplified, and recorded on analog tape. At depths greater than 70m, nearly isovelocity conditions prevailed with a sound speed of ~1421.7m/sec. We used the scattering from the 22-kHz signal to measure time-of-flight and verify alignment and slant ranges. A 1.12-m diameter, hollow, steel spherical target was used to test our ability to measure target strength (TS). Assuming a perfectly reflected surface, this sphere possessed a theoretical TS of -11.1dB at 5 kHz; the measured TS was -12.3dB.

Bubble clouds were produced at 91.4-m depth using a pressurized steel enclosure, vented to the lake via a concentric circular array of 48 22-gauge hypodermic needles connected to a bank of 24 solenoid valves. A 3.5-sec burst of air at 68.95kPa over-pressure yielded a roughly cylindrically shaped cloud length  $\approx 1.7\pm0.3$ m, radius  $\approx 0.24\pm0.01$ m, with a void fraction of  $\approx 0.0040\pm0.00007$ . The bubble sizes, which were measured with a video camera at depth as well as in a pressurized tube, were normally distributed about a peak at

1.1±0.2mm radius, which corresponds to  $f_0 \approx 9.2\pm 1.7$ kHz at a depth of 87.6m.

The clouds rose at  $\approx 0.3$ m/sec, yielding a time-varying echo that reached a maximum level as the cloud crossed the range axis. Rms voltages were computed for both the main bang signal (mv) and for each echo return (ev). These values where inserted, along with the transmission loss factor (TLF), into the following expression for the target strength:  $TS = 20 \log[ev / mv]$ ;  $TLF = TL_{sr} - TL_{sr}$ ; where the TLF accounts for source-receiver, source-target, and target-receiver transmission losses, assuming spherical spreading of the propagating waves. Target strength measurements obtained using the parametric sources are plotted in Fig. 2. The target strengths are high (TS = -.31 dB) near 300 Hz. Also note the two intense peaks near 300Hz and the single peak near 1.3kHz. These resonant peaks if due to single bubbles would require bubbles of 3.4cm and 7.8mm radii. Bubbles of this size were simply not observed.

It is probable that these peaks are the result of resonance scattering due to collective oscillations of the bubble cloud. The resonance frequency for a spherical bubble cloud is given by equation the Minaert equation (Eq. 8). We assume that the spherical cloud which has the same volume as the cylindrical cloud will describe the resonance frequency. Such a cloud, with  $\chi=0.0055$  and  $r_0=0.38 m$ , has an isothermal resonance frequency of 324±53Hz, very close to our measured value of 325Hz. Moreover, the target strength is given by the

definition of 
$$A_0$$
 in equation 6:  

$$\frac{I_s}{I_s} = \left| \frac{P_s}{P_s} \right|^2 = A_o \cdot A_o^* / k^2 \quad ; \quad r = 1m$$

where  $I_s$  and  $I_i$  are the scattered and incident intensities respectively. For the cloud in question, this yield a calculated TS of -3.9 $\pm$ 0.8dB at approximately 309Hz. Although the measurements and calculations of the fundamental resonance frequency of the bubble cloud agree to within the stated uncertainty, the calculated target strengths are too low (see Fig. 2 However clearly shown in Fig. 2 is an interference pattern near 300Hz. We modeled this interference and found that our measured target strength was -3.4 $\pm$ 0.9dB, excellent agreement. Parametric and conventional measurements exhibit "reasonably" good agreement despite the fact that reverberation made it difficult to obtain conventional source data for frequencies less than 1kHz.

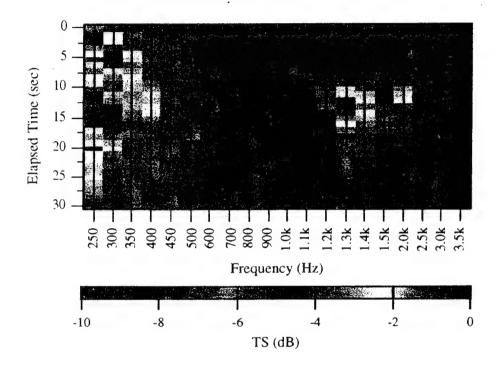


Figure 2. The TS versus time and frequency for the Seneca Lake scattering experiment.

#### 6. CONCLUSION

Acoustically-compact, bubbly regions (clouds) can behave as monopole radiators and scatters of sound. The application of classical theory coupled with the low frequency mixture properties produces an analogous single-bubble description of the bubble cloud in which the resonance frequency of the volume mode is described by a modified Minnaert formula and a simplified monopole expression exists for the backscatter target strength. Thus micro-bubble clouds produced near the sea surface are important in the production and scattering of low frequency sound.

### 7. REFERENCES

- 1. W. Carey, R. Roy, Ocean Reverberation, D. Ellis et al. (eds.), Kluwer (1993) 25.
- 2. W. Carey, J. Fitzgerald, Natural Physical Sources of Underwater Sound, B. Kerman (ed)., Kluwer (1993) 277.
- 3. W. Carey, D. Browning, Sea Surface Sound, B. Kerman (ed.), Kluwer (1988) 361.
- 4. W. Carey, J. Fitzgerald et al., J. Acoust. Soc. Am., 93 (1993) 3178.
- 5. R. Roy, W. Carey, et al, J. Acoust. Soc. Am., 92 (1992) 2993.

## Appendix E

In <u>Proceedings of the 2nd European Conference on Underwater Acoustics</u>, ed. by L. Bjorno, Pub. by European Commission, Luxembourg, 195-202 (1994).

Can near-surface bubble clouds and plumes lead to anomalous perturbations in low-frequency sea-surface scattering?

R. A. Roy<sup>a</sup>, J. A. Schindall<sup>b</sup>, W. M. Carey<sup>c</sup> and L. A. Crum<sup>a</sup>

<sup>a</sup>Applied Physics Lab, Univ. of Washington, 1013 NE 40th St., Seattle, WA 98105, USA bDept. of Physics, Univ. of Mississippi, University, MS 38677, USA <sup>c</sup>Advanced Research Projects Agency, 3701 North Fairfax Drive, Arlington, VA 22203, USA

This paper addresses the importance of near-surface bubble scattering using classical mixture theory and the method of images. This analysis shows that under the correct conditions, clouds of bubbles near the surface of the sea can produce point-like scattering as observed in several recent experiments.

### 1. INTRODUCTION

Studies indicate that microbubble layers, plumes and clouds are produced when waves break and that these assemblages are convected to depth by Langmuir circulation [1]. The commonly held view is that these persistent bubbly assemblages can act as efficient scatterers of sound. Recent tests of low-frequency active sonar systems (Gulf of Alaska, 1992) indicate the presence of prominent, near-surface targets during conditions of high sea state [2]. Those investigators concluded that prominent returns in the 800-1200 Hz range were due to bubble clouds with lifetimes less than 45 seconds. At the same test, Adair *et al.* [3] also observed strong returns (target strength > -5 dB) from seemingly short-lived targets positioned within a few meters of the surface. Models have been developed that describe the weak scattering from deep microbubble clouds (so-called tenuous bubble clouds) with void fractions,  $\beta$ , between  $10^{-9}$ . However, such clouds persist for more than 45 seconds and are probably not the source of these bright, point-like targets (see ref. 4 for a review).

It is more likely that these short-lived, near-surface targets result from bubble plumes generated beneath plunging breakers. Such plumes possess length scales of up to 1 m, void fractions of order 10<sup>-3</sup>, and lifetimes on the order of seconds [5,6]. If the plumes detach, there exits a three-dimensional, high void fraction bubble cloud momentarily submerged at a depth of order 1 meter, or possibly even more. It has been our contention that such a cloud could exhibit resonance scattering properties that translate to high target strengths at low frequencies for short periods of time. This hypothesis is based on the classical theory that when a large number of bubbles occupy an acoustically compact region, coupled oscillations result, and the propagation of sound is determined by the properties of the mixture; the free gas establishes compressibility and the water provides inertia [4,5,7,8,9].

Since the burden of proof lies with the experimentalist, a lake test was carried out in the fall of 1992 [10]. This test, which was performed at the NUWC Lake Seneca test facility, involved the production of a bubble cloud of known geometry and void fraction at a depth of 89 m beneath the surface of the lake. Measurements of the low-frequency target strength (TS) show a monopole resonance whose level, -3 dB, and frequency, 325 Hz, are well predicted by classical scattering theory described below. The generalization of this model to near-surface conditions is the topic of this paper.

### 2. THEORETICAL MODEL

2.1 Scattering from bubble clouds in the free field

We now follow the approach of Morse and Ingard [11] (see Ref. 4) in which we assume a plane wave incident on a compliant sphere of radius  $R_c$  surrounded by a continuous medium

of density and sound speed  $\rho$  and c respectively. Likewise, we consider the spherical bubble cloud to be a homogeneous, dispersive medium characterized by an *effective* density and wave number given by [12],

$$\rho_{eff} = \beta \rho_{air} + (1 - \beta)\rho , \qquad k_{eff}^2 = k^2 + \frac{4\pi\omega^2 a\eta}{\omega_o^2 - \omega^2 + 2ib\omega} , \qquad (1)$$

where k is the wave number in the liquid, a is the radius of individual bubbles (considered to monodispersed in size),  $\eta$  is the number of bubbles per unit volume,  $\omega_o$  is the resonance frequency of the bubbles,  $\beta$  is the void fraction, and b is the complex damping constant. For  $\omega << \omega_o$ , one can show that the real component of the complex phase speed is given by,

$$c_{eff}^2 = \frac{\kappa P_{amb}}{\kappa P_{amb} + \beta \rho} . \tag{2}$$

Here,  $P_{amb}$  is the ambient pressure and  $\kappa$  is the polytropic exponent. Note that  $c_{eff}$  decreases with increasing void fraction. Indeed, for  $\beta$  greater than approximately  $10^{-2}$ , the effective sound speed of the mixture is less than that encountered in either air or water alone.

For the scattering problem in an infinite medium we solve the wave equation subject to the boundary conditions of continuous pressure and normal velocity. The far-field solution is a superposition of incident plane waves and scattered spherical waves, the latter of which are represented by an expansion in Legendre polynomials given by

$$p_{s}(r) = i \frac{|p_{i}|}{2k} \frac{e^{ikr}}{r} \sum_{m=0}^{\infty} (2m+1)(1+C_{m}) P_{m}(\cos \theta_{s})$$
 (3)

The coefficients  $C_m$ , which are given in refs. 4 and 11, depend on the material properties of the sphere. For acoustically compact scatterers  $kR_c < 1$ ; the monopole term in the expansion dominates and the far-field backscattered pressure is approximated by

$$p_{s} \approx i p_{o} \frac{k^{2} R_{c}^{3}}{3r} \frac{\left(1 - \frac{\rho c^{2}}{\rho_{eff} c_{eff}^{2}}\right)}{\left(1 - \frac{\rho c^{2}}{\rho_{eff} c_{eff}^{2}} \frac{(kR_{c})^{2}}{3}\right) - i \left(\frac{\rho c^{2}}{\rho_{eff} c_{eff}^{2}} \frac{(kR_{c})^{3}}{3}\right)} . \tag{4}$$

The second term in the denominator accounts for radiation damping while the first term, when set equal to zero, defines a resonance condition. This results in a monopole resonance frequency and scattered pressure amplitude given by

$$\Omega_o = \frac{1}{R_c} \sqrt{\frac{3\gamma P_{amb}}{\beta \rho}} \quad \text{and} \quad \left| \frac{p_s}{p_i} \right| = \frac{1}{k_o r} \left| \frac{\rho_{eff} c_{eff}^2}{\rho c^2} - 1 \right| .$$
(5)

Note that the expression for  $\Omega_o$  has the same form as the Minneart equation defining the monopole resonance frequency of a spherical bubble. To obtain the free-field backscattered target strength  $(TS_{ff})$  we substitute either Eq. 3 or, for low-frequency scattering, Eq. 4 into

$$TS_{ff} = 10\log\left(\frac{I_s}{I_i}\right)_{r=1m} = 20\log\left|\frac{p_s}{p_i}\right|_{r=1m}.$$
 (6)

Table 1 compares predicted resonance frequency and target strength with measurements obtained at Lake Seneca  $(R_c \cong 0.38 \,\mathrm{m}; \beta \cong 5.5 \times 10^{-3}; \mathrm{submerged \, depth} \cong 89 \,\mathrm{m})$ . Since our

concern is with the monopole resonance response, we calculate only the m=0 term in Eq. 3. The good agreement between theory and measurement supports our view that an effective media approximation is a reasonable way to model low-frequency bubble cloud scattering; indeed, it appears that one can predict  $\Omega_o$  and resonance  $TS_{ff}$  using the simple algebraic expressions given in Eq. 5. It is important to note that we observe agreement despite the fact that the Seneca cloud was spheroidal rather than spherical, the reason being a monopole response corresponds to a *volume-dependent*, breathing mode oscillation. For an acoustically compact scatterer, the monopole resonance behavior is well approximated by a sphere of equal volume. In the case of the Seneca cloud, this corresponded to a sphere 0.38 m in radius.

Table 1
Monopole resonance frequency and target strength for the Lake Seneca bubble cloud.

	Eq. 3 $(m = 0)$	Equation 5	Measured	
$\Omega_{_{o}}$	$326 \pm 25 \text{ Hz}$	$294 \pm 25 \text{ Hz}$	$325 \pm 25 \text{ Hz}$	
$TS_{ff}(\omega = \Omega_o)$	$-2.9 \pm 0.7 \text{ dB}$	$-3.0 \pm 0.7 \text{ dB}$	$-3.4 \pm 0.9 \text{ dB*}$	

<sup>\*</sup>The measured value of  $TS_{ff}$  was obtained at 300 Hz, and not at  $\Omega_o$ .

### 2.2. Scattering from bubble clouds near the ocean surface

At low frequencies we model the cloud as a monopole scatterer positioned a distance d beneath a plane horizontal surface characterized by a reflectivity coefficient  $\mu$ , which can vary between -1 and 0. We assume incident plane waves and far-field propagation. The method of images yields the following expression for the near-surface, backscatter target strength [13]:

$$TS_{ns} = TS_{ff} + 10\log\{1 + \mu^2 + 2\mu - 4\mu\sin^2(kd\sin\theta_g)\}^2 , \qquad (7)$$

where  $\theta_g$  is the grazing angle measured relative to the horizontal. The amplitude reflection coefficient describes the roughness of the sea surface;  $\mu = -1$  corresponds to a flat surface, while  $\mu = 0$  corresponds to a very rough surface. We can relate  $\mu$  to U, the wind speed measured at a reference height [5], using the Kirchoff approximation for the Pierson-Moskowitz sea surface for a specific set of boundary layer conditions one has [14]

$$\mu \approx -e^{-2(kH\sin\theta_{\rm g})^2}$$
,  $H^2 = \frac{\alpha_1 U^4}{4\alpha_2 g^2}$  (8)

where H is the rms wave height derived from the Pierson-Moskowitz wave height spectrum, and where  $\alpha_1 = 8.1 \times 10^{-3}$ ,  $\alpha_2 = 0.74$ , and g = 9.81 m/s<sup>2</sup>. At low frequencies or low wind speeds the surface appears acoustically flat and  $\mu \to -1$ , as it should. Similarly, at high frequencies or high wind speeds the surface appears acoustically rough, which disrupts coherent surface reflection yielding free-field acoustic behavior (i.e.  $\mu \to 0$ ).

It is evident from Eqs. 5, 7 and 8 that the resonance backscatter target strength for a near-surface bubble cloud is a complex function of the various parameters. From Eq. 5 we see that increasing either  $R_c$  or  $\beta$  reduces  $\Omega_o$  and increases  $TS_f$ . However, the proximity of the surface is parameterized by kd, i.e., relative to the acoustic wavelength. For kd << 1 and  $\mu = -1$ , the scattered pressure scales with  $(d/\lambda)^4$ . From this we see that, although large high- $\beta$  clouds possess significant cross sections at resonance, they are also subject to substantial surface image interference owing to their close proximity (both oceanographically and acoustically) to the surface. It would appear that there are optimum combinations of physical and environmental parameters leading to significant target strengths (say, larger than -10 dB).

### 3. RESULTS AND DISCUSSION

At this point we consider a "case study", in which we explore the scattering characteristics of a 5-cm radius cloud where the grazing angle is fixed at  $10^{\circ}$ . Within the bounds of our simple model, the remaining parameters influencing the scattering characteristics of this cloud are its void fraction, depth, and wind speed. We will consider both free-field and near-surface scattering. All calculations of  $TS_{ns}$  are computed using the m=0 term from the Eq. 3; thus, we consider *only* low-frequency (i.e. monopole) backscattering behavior.

Shown is Fig. 1a is the target strength versus frequency for this cloud resident at a depth of 2 m and possessing a void fraction of  $5 \times 10^{-3}$ . Note the monopole resonance peak at 786 Hz, where  $TS_{ff}$  is -9.5 dB. A wind speed of 20 m/sec does little to modify the response, for the surface is very rough. However, lowering the wind speed to 7 m/sec raises  $TS_{ns}$  to about 0 dB. This is due to constructive interference between the real and image scatterers as illustrated in the plot of resonance  $TS_{ns}$  versus depth show in Fig. 1b. The "Lloyd's Mirror" interference pattern is clearly the cause of the elevated target strength.

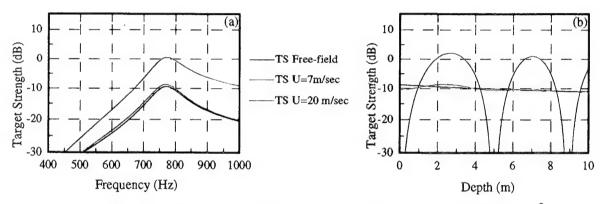


Figure 1. Low-frequency backscatter from 5-cm radius cloud for  $\beta = 5 \times 10^{-3}$  and  $\theta_g = 10^\circ$ : (a) depth = 2.0 m; (b) frequency = 786 Hz.

The global scattering properties of this cloud are illustrated in the surface plots of Fig. 2. Figure 2a is a plot of the  $TS_{ff}$  at resonance versus depth and void fraction. As predicted by Eq. 5, increasing the void fraction results in a significant increase in  $TS_{ff}$ , and a decrease in  $\Omega_o$ . Note also the weak dependence on submerged depth. Figures 2b through 2d depict the influence of the surface. The surface-dipole interference pattern is most evident at 7 m/sec, where  $\mu \cong -1$  over the entire frequency range. Target strengths as high as +10 dB are observed for clouds with void fraction  $5 \times 10^{-2}$  positioned at depths ranging from 7-8 m. However, such clouds are not deemed likely to exist in the open ocean.

A sequential consideration of Figs. 2b through 2d serves to illustrate the effect of increasing wind speed. As the wind picks up, the surface become rougher and therefore less influential acoustically. Indeed, it appears that at U = 20 m/sec, free-field conditions prevail for all but the lowest resonance frequencies (i.e. highest void fractions). The notion of an optimal combination of void fraction and depth is clearly evident in these plots, which exhibit

pronounced peaks and valleys as these parameters vary.

Consider now the role of increasing bubble cloud size. Figure 3a shows a plot of  $TS_{ff}$  versus frequency for clouds with radii ranging from 5 to 50 cm. All four clouds possess a void fraction of  $5 \times 10^{-3}$ . As expected, increasing cloud size is equivalent to decreasing the resonance frequency and increasing the target strength in the free field. Conversely, Fig. 3b shows  $TS_{ns}$  for the same clouds at a depth of 2 m. Now, it's the *smaller* clouds that exhibit the higher target strengths at resonance. In general, this is due to the fact that smaller clouds resonate at higher frequencies, which means that the surface is further removed (acoustically

speaking) and therefore less influential. However, in this particular instance, the contribution of constructive interference between the real and image scatterers plays a role as well. Regardless, we should not be too quick to ascribe free-field scattering properties to bubble clouds near the surface. Sometimes, it's the smallest clouds that make the largest contribution.

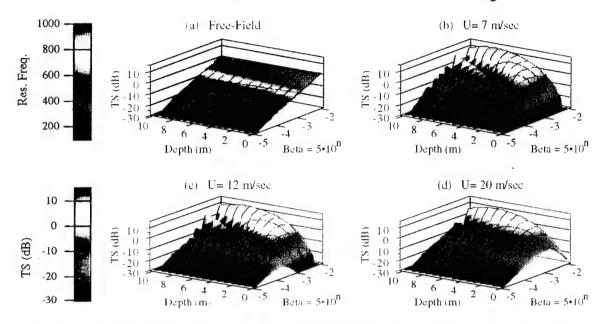


Figure 2. Monopole resonance backscatter from 5-cm radius cloud for  $\theta_g = 10^\circ$ . The color scale denotes (a): monopole resonance frequency; (b) through (d): TS at resonance.

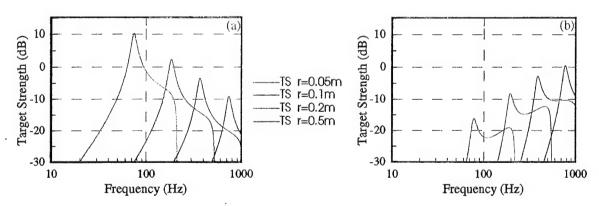


Figure 3. Low-frequency backscatter from various clouds for  $\beta = 5 \times 10^{-3}$  and  $\theta_g = 10^\circ$ : (a) free-field calculations; (b) depth = 2.0 m and U = 7 m/s.

### 4. SUMMARY AND CONCLUSIONS

In this study, we have developed a simple model describing the low-frequency scattering properties of high- $\beta$  bubble clouds in both the free field and near the ocean surface. This model, which is based on an effective medium approximation and acoustically compact, spherical scatters, successfully predicts the results of a bubble cloud scattering experiment carried out at Lake Seneca [10]. The introduction of the surface is facilitated by the method of images and is subject to the same low-frequency constraint imposed by the compact scatterer assumption. This model is not intended to serve as a exact representation of oceanic bubble

cloud scattering. We believe that such an endeavor offers limited utility, for the accuracy of any model is ultimately bounded by the accuracy of the input data. The model is kept simple by design, for only then can we express complex physical behavior in simple analytical form. Simple analytic expressions facilitate the exploration of parameter space and, more importantly, serve to illuminate the underlying physics.

Densely populated bubble clouds appear to resonate somewhat like a free bubble (see Eq. 5). Increasing the size of the cloud decreases the resonance frequency and increases the free-field target strength at resonance. Increasing the void fraction has the same effect due to the reduction in sound speed within the cloud. For example, a 0.5-m cloud ( $\beta = 5 \times 10^{-3}$ ) resonates at 75 Hz and possesses a  $TS_{ff}$  of +10 dB at resonance, whereas a 0.1-m cloud with

the same void fraction resonates at 370 Hz and has a target strength of -3.5 dB.

The introduction of the surface leads to a dipole characteristic which can either increase or decrease the target strength depending on the frequency and the depth. Since the resonance frequency is also a function of cloud size and void fraction, the manifold of possible behavior is quite broad. In general, larger clouds are more affected due to the fact that they resonate at lower frequencies; the proximity of the surface is defined relative to  $\lambda$ . Surface roughness reduces this *image interference effect*. As either the wind speed or the acoustic frequency increases, the environment asymptotes to that of a free field.

A case study of a 5-cm radius cloud revealed target strengths as high as +10 dB at depths exceeding 7 m and for a wind speed of 7 m/sec. For a cloud with  $\beta = 5 \times 10^{-3}$  at a depth of 2 m, the calculated resonance frequency and target strength was 786 Hz and 0 dB respectively. It is important to recognize that, given existing oceanographic data, the notion of such a cloud temporarily residing at a depth of 2 meters beneath a breaking wave event is not unreasonable. A 0-dB target strength is quite significant, and we conclude that these small, high void fraction clouds may indeed be the source of the short-lived, near-surface scattering features observed by investigators such as Gauss et al. [2] and Adair et al. [3].

### 4. ACKNOWLEDGMENT

We wish to acknowledge the technical contributions of M. Nicholas, helpful discussions with A. Prosperetti, James Fitzgerald and P. Dahl, and the financial support of the Office of Naval Research (Ocean Acoustics).

#### 5. REFERENCES

1. S. Thorpe, Phil. Trans. Roy. Soc. A, 304 (1982) 155.

2. R. Gauss, J. Fialkiowski, and R. Soukup, Ocean Reverberation, D. Ellis et al. (eds.), Kluwer (1993) 235.

3. R. Adair, M. Huster, and D. Miklovic, J. Acoust. Soc. Am., 92 (1992) 2479.

- 4. W. Carey and R. Roy, Ocean Reverberation, D. Ellis et al (eds.), Kluwer (1993) 25.
- 5. W. Carey and J. Fitzgerald, Natural Physical Sources of Underwater Sound, B. Kerman (ed.), Kluwer (1993) 277.

6. E. Monahan, Sea Surface Sound, B.R. Kerman (ed.), Kluwer (1988) 503.

7. L. d'Agostino & C. Brennen, J. Acoust. Soc. Am., 84 (1988) 2126.

8. W. Carey and M. Browning, Sea Surface Sound, B. Kerman (ed.), Kluwer (1988) 361.

9. A. Prosperetti, J. Acoust. Soc. Am., 84 (1988) 1042.

- 10. R. Roy, W. Carey, M. Nicholas, J. Schindall, and L. Crum, J. Acoust. Soc. Am, 92 (1992) 2993.
- 11. P. Morse and K. Ingard, Theoretical Acoustics, McGraw Hill (1968) 413.

12. K. Commander and A. Prosperetti, J. Acoust. Soc. Am., 85 (1989) 732.

13. L. Crum, J. Schindall, R. Roy, and W. Carey, Proceedings: The 1993 NATO Low-Frequency Active Sonar Symposium, Kluwer (in press).

14. E. Thorsos, J. Acoust. Soc. Am., 88 (1990) 335.

## Appendix F

# Acoustic Scattering from Compact Bubble Clouds

by

# Jeffrey A. Schindall, B.S. Computer Science B.S. Mathematics

### Dissertation

Presented to the Faculty of the Graduate School of

The University of Mississippi

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Mississippi

December 1995

### Acoustic Scattering from Compact Bubble Clouds

Jeffrey A. Schindall, Ph.D.

The University of Mississippi, 1995

Supervisor: Ronald A. Roy

In this study, a simple model describing the low–frequency scattering properties of high void fraction bubble clouds in both the free field and near the ocean surface is developed. This model, which is based on an effective medium approximation and acoustically compact scatters, successfully predicts the results of the bubble cloud scattering experiment carried out at Lake Seneca in New York state for frequencies consistent with the model assumptions (Roy et al., 1992). The introduction of the surface is facilitated by the method of images and is subject to the same constraint of low–acoustic frequency imposed by the compact scatterer assumption. This model is not intended to serve as an exact replicate of oceanic bubble cloud scattering. The model herein was kept simple by design, for only then can the complex physical behavior be expressed in a simple analytical form. Simple, analytic theories facilitate the underlying physics.

### Contents

Ackno	wledgments	iii
Abstract		
List of	Tables	ix
List of	Figures	x
Chapte	er 1 Introduction	1
1.1	Ambient noise in the ocean	3
1.2	The Theory of Low Frequency Ocean Ambient Noise: A Paradigm	
	Shift	8
	1.2.1 Experimental evidence of low-frequency noise produc-	
	tion by bubble clouds	11
1.3	Breaking Waves, Bubble Clouds and Low-Frequency Ambient	
	Noise	15
1.4	Sea-surface scatter	22
1.5	Sea surface scattering: selected field experiments	27
1.6	Sea surface scattering: modeling efforts	32
1.7	Resonance scattering from bubble clouds	34
	coustic scattering from bubble clouds in the ab-	
sence	of boundaries: Theory	38
Chapte	er 2 Propagation of sound in a bubbly mixture	39
2.1	Introduction	39
2.2	Sound speed in bubbly liquids — Low Frequencies	41
2.3	Sound speed in a bubbly medium — dispersion	44
	2.3.1 Theory	45
	2.3.2 Linearized single—bubble dynamics	47

	2.3.3 Effective wave equation	50
	2.3.4 Calculations and experimental results	52
	2.3.5 Bubble size distributions	60
2.4	Discussion and Summary	65
Chapte	er 3 Acoustic scattering from a spherical bubble clouds	:
	eoretical model	69
3.1	Introduction	69
3.2	Formalism	72
3.3	Backscattering form function	80
	3.3.1 Case Studies	85
3.4	The low–frequency limit	90
3.5	Discussion and Summary	97
	coustic scattering from bubble clouds in the ab-	
sence	of boundaries: Experiment	100
Chapte	er 4 The Lake Seneca experiment: Project overview and	l
	oble cloud generation	101
4.1	Introduction	101
	4.1.1 Sound speed profile	105
4.2	Bubble maker deployment and operation	105
4.3	Bubble cloud characterization	112
	4.3.1 Bubble cloud rise time	113
	4.3.2 Bubble cloud rise velocity	115
	4.3.3 Bubble cloud length	115
	4.3.4 Bubble cloud shape and volume	116
	4.3.5 Effective bubble cloud radius	119
	4.3.6 Total gas volume measurement	119
<b>.</b>	4.3.7 Bubble radius distribution measurements	123
4.4	Summary	121
Chapte	er 5 The Lake Seneca experiment: acoustic backscattering	
mea	abdicincing dira diracjoso	<b>129</b>
5.1	The experimental test plan	129
	5.1.1 The parametric source	131
	5.1.2 Bubble cloud generation	133
5.2	The measurement of target strength	137
5.3	Test range layout	138

5.4	Signal conditioning and recording	142
5.5	Test range calibration	148
5.6	"On-line" data processing	151
5.7	Post–experimental analysis	158
	5.7.1 Digitizing the data tapes	158
	5.7.2 Data analysis	163
TTT T	Dealeasttoring from accustically compact hubble	
	Backcattering from acoustically compact bubble s near the sea surface	173
Chante	er 6 Acoustic backscattering from compact bubble cloud	e
	eath the ocean surface	174
6.1	Introduction	174
	6.1.1 Prominent, near—surface scattering features	176
6.2	Theoretical model	178
	6.2.1 Scattering from bubble clouds in the free field	178
	6.2.2 Scattering from bubble clouds near the ocean surface .	181
	6.2.3 The method of images	181
	6.2.4 The Kirchoff approximation	184
6.3	Results and discussion	185
6.4	Summary and conclusions	189
Chapte	er 7 Concluding remarks	192
7.1	Collective bubble oscillations	194
7.2	Scattering	195
7.3	The burden of proof	196
7.4	Scatter from subsurface bubble clouds	197
7.5	Remarks	199
Appen	dix A Target Strength	201
A.1		201
	TS measurements — Steady state	204
	TS measurements — transient pulses	
Appen	dix B Seneca-2 test schedule and recording parameters	215
Appen	dix C Seneca-3 test schedule and recording parameters	220

Appendix D Error analysis methods	224	
D.1 Directly measured quantities	224	
D.2 Derived quantities		
D.3 Experimental Uncertainties	226	
Appendix E Rise-time dependent target strength measurement	s <b>23</b> 0	
Appendix F Source Codes		
Bibliography		

### List of Tables

$1.1 \\ 1.2$	Beaufort scale of wind force and sea state	4
	event	17
2.1	Physical parameters used in this section	50
2.2 2.3	Comparison of sound speed formulations	53
	respectively	62
4.1	Cloud generation parameters	110
4.2	Radius versus height profile of the Lake Seneca bubble cloud	120
4.3	Statistics from in situ BSD measured slightly above a single	105
4.4	needle tip at Lake Seneca	125
	in situ BSD to a triple–Gaussian curve	126
4.5	Lake Seneca experimental precision analysis	128
5.1	Frequency dependent source level and beam width for the parametric source. These measurements were performed by tilting	
	the PS about the vertical and using receiver H1	134
5.2	Range layout and transmission loss factors	139
5.3	Tape deck (Teac SR31–DF) specifications	145
5.4	Monopole resonance frequency and target strength for the Lake Seneca bubble cloud.	171
B.1	Seneca-2 Data Runs and Recording Parameters	216
C.1	Seneca-3 Data Runs and Recording Parameters	221

## List of Figures

1.1	Ambient noise spectrum and source mechanisms. From Wenz (1962). The levels in this figure should be increased by 66 dB	
	to conform to the current standard (dB re $1 \mu \text{Pa}$ )	5
1.2	Urick and Hoover's windspeed and grazing angle dependent	
1.4	60 kHz surface backscattering data	25
1.3	Summary of SUS surface scatter results	28
1.4	Sea surface scattering measurements in the upper 5 m of the ocean surface during CST-7. In the figure to the left, the wind speed was $U = 4.1 \text{ m/s}$ (significant wave height 2.2 m, no breaking waves). The average scattering strength in this image is -53 dB. Only a few events, believed to be salmon, exceed the average by +10 dB. In contrast, in the figure on the right, the wind speed is $U = 16.8 \text{ m/s}$ (wave height 5.0 m) and a great number of high target strength scattering events exceeding the average (-30 dB) by +10 dB are prevalent. The image resolution is 1 m in range and 0.005 radians in azimuth (120-1430 Hz). This figure	
	was graciously provided by Aretè Associates (Miklovic et al., 1995)	31
2.1	Sound speed versus $\beta$	45
2.2	Polytropic index versus frequency	50
2.3	Phase speed and attenuation versus frequency	54
2.4	Phase speed versus frequency	55
	Frequency dependent phase speed. The bubble radius was 1.11 mm	
2.5	and the void fraction $\beta = 10^{-2}$ . The crosses represent the data	)
	measured by lowering a hydrophone down the length of the	
	measured by lowering a hydrophone down the length of the	
	standing wave tube, and the circles using the LDV. Data cour-	58
	tesy of Cheyne et al. (1995)	90

2.6	Phase speed as a function of frequency for a discrete bubble radius distribution. The average void fraction is $\beta = 10^{-3}$ , and the bubble distribution is listed in Table 2.3.	63
2.7	the bubble distribution is listed in Table 2.3	03
2.8	and the limits of the distribution is $30 \mu\mathrm{m} \leq a \leq 240 \mu\mathrm{m}$ . Attenuation and phase speed as a function of frequency for a truncated Gaussian bubble radius distribution. The void fraction is $\beta = 10^{-4}$ ; the mean bubble size is $R_o = 60 \mu\mathrm{m}$ ; the standard deviation is $\sigma_{R_o} = 60 \mu\mathrm{m}$ ; and the range of the distribution are $30 \mu\mathrm{m} \leq a \leq 240 \mu\mathrm{m}$ .	<ul><li>63</li><li>64</li></ul>
3.1	Scattering geometry. The z-axis is aligned with the direction of plane wave propagation. The coordinate system origin is	
	located at the center of the sphere.	75
3.2	Backscattering form function versus $ka$ for an impenetrable sphere	86
3.3	Backscattering form function versus $ka$ for a bubble	87
3.4	Backscattering form function versus $ka$ for a bubble cloud	88
3.5	TS versus frequency	90
3.6	Modes of sphere	91
3.7	A comparison of the asymptotic form function with the exact	02
3.8	Peak monopole TS and frequency as a function of void fraction	93
	for cloud radii $a = 0.1$ m and $a = 0.25$ m using the exact series expansion (solid) and the asymptotic expression (dashed)	95
3.9	Peak monopole TS and frequency as a function of cloud radius	
	for a void fraction of $\beta = 10^{-4}$ and $\beta = 10^{-3}$ using the exact	0.0
	series expansion (solid) and the asymptotic expression (dashed).	96
4.1	Sound speed profile at Lake Seneca test site	106
4.2	Photograph of the bubble maker	107
4.3	Schematic of the Seneca Lake bubble maker	108
4.4	Pressure difference $(\Delta P)$ between the lake and the bubble maker	
	at a depth of 91.6 m as a function of time during a cloud release.	
	The cloud was released at $t = 0$ at which point the internal	
	pressure decreased. The electronic regulator was set to a 0.5 psi	
	threshold at which point the input solenoid was opened allowing	
	air to enter the bubble maker.	111
4.5	Video rig used to measure physical parameters of cloud at depth.	114

4.6	Mosaic of a bubble cloud. A series of snapshots were taken di-	
	rectly from the video monitor during tape playback. The images	
	to the left and to the right were not taken from the same cloud.	
•	However, the sequence of photos in each column were obtained	
	from a single rising bubble cloud. In each column, each con-	
	secutive snapshot is separated in time by approximately 1.4 s.	
	Notice the variation in bubble population density. The dark	
	vertical stripe in the left portion of the image is the suspension	
	cable used to lower the bubble maker. Thus it is clear that the	
	bubble cloud rose slightly off axis due to a slight current in the	
	lake.	118
4.7	Frustum of a cone. The total volume of the bubble cloud was	
	determined by accumulating 31 frustum volumes using the ra-	
	dial measurements in Table 4.2	119
4.8	Digitized flow meter output as a function of time during a cloud	
	release. The output was integrated and scaled to yield the total	
	air volume injected into the cloud, nominally $1.22 \pm .014  l.$	121
4.9	Bubbles released from single needle tip as viewed from the sub-	•
	mersible video camera. The scaled ruler (mm gradations) is	
	visible on the right. The bright spot in the center is due to the	
	intense lighting needed to illuminate the rising bubbles	124
4.10	Bubble size distribution measured just above a single needle tip	-
	at Lake Seneca. The curve represents a triple-gaussian non-	
	linear least squares fit to the measured data. For phase speed	100
	calculations the fit is truncated between 0.1 mm and 3.2 mm.	126
4.11	Effective phase speed versus frequency for the Lake Seneca bub-	
	ble cloud using monodispersed (solid line), discrete (jagged-	
	dashed line), and Gaussian (smooth-dashed line) bubble size	
	distributions. Here the void fraction is $\beta = 0.55 \times 10^{-2}$ , and	
	$P_{\infty} = 9.82 \times 10^5 \mathrm{Pa}$ . For all frequencies of interest in this ex-	
	perimental analysis (i.e., below 2000 Hz, the sound velocity in	128
	the mixture asymptotes to 460 m/s	120

5.1	Layout of the test range. The receivers in the backscatter di-	
	rection are denoted by H1, H2, and H3. Hydrophone H4 is in	
	the forward direction and was used primarily to measure noise	
	emissions from newly formed bubble clouds (Nicholas et al., to	
	be submitted). H3 and H1 were also used as a conventional	
	sources (CS1 and CS2, respectively) during some of the tests.	
	The PS beamwidth is indicated approximately by the region	
	between the solid lines	130
5.2	Typical parametric source beam pattern	133
5.3	Photograph of the parametric source	135
5.4	Scattering geometry for a single hydrophone and the target not	
	in the backscatter position. $R_{SH}$ is the distance between the	
	source and the receiver; $R_{ST}$ is the distance between the source	
	and the target; and $R_{HT}$ is the distance between the target and	
	the hydrophone. When the target is in the backscatter position,	
	$R_{ST} = R_{HT} + R_{SH} \dots \dots$	140
5.5	Total transmission loss factor for receiver H1 as a function of	
	bubble cloud rise-time relative to cloud release. The dashed line	
	corresponds to the total transmission loss correction applied to	
	the Seneca-3 data, and the solid line for the Seneca-2 data.	
	When the cloud is in the backscatter position, $t \approx 11 \mathrm{s}$ , the	
	TLF corresponds to the values given in Table 5.2	142
5.6	Instrumentation setup for a single hydrophone during the Seneca-	
	3 experiment. The large amplitude $\approx 22\mathrm{kHz}$ primaries from the	
	parametric source were removed using the 2 kHz passive low-	
	pass filter. Then the signal was split into two channels using a	
	"T" and fed into a pair of pre-amplifiers and active low-pass	
	filters prior to recording to individual main bang and echo chan-	
	nels on the reel-to-reel tape deck. The pre-amplifier gain was	
	set to optimally amplify either the incident or scattered echo	
	pulse respectively in a $1 V_{peak}$ range prior to recording. Dur-	
	ing the Seneca-2 experiment, the main bang and echo signals	
	were recorded from separate data runs, thus the "T" was not in-	
	serted. Online processing was performed by monitoring narrow	
	band passed signals on a LeCroy 9450 digital oscilloscope	144

5.7	Sample reel-to-reel tape segments (elapsed time) for the Lake Seneca experiments. The tape deck was capable of recording	
	7 data tracks simultaneously and the signals recorded to each	
	track is listed above. The signals recorded and preamplifier gain	
	settings used are indicated here only for tracks 1-4. Signals were	
	recorded to tape for approximately 60 s per cloud release. In the	
	region between data runs a 2s calibration tone was recorded.	
	Note: Due to the instrumentation setup used in the Seneca-2	
	effort, a single main bang recording was conducted prior to a	
	series of cloud releases. During the Seneca-3 experiment, the	
	main bang and echo recordings were made simultaneously	147
5.8	Photograph of instrumentation setup. Located on the top of	
0.0	the equipment rack is a digital voltmeter used to monitor the	
	internal over pressure of the bubble maker. Beneath this, the	
	Deep Thought bubbler controller unit is mounted in the upper	
	most location of the equipment rack. A signal generator and the	
	LeCroy 9450 digital oscilloscope are next, followed by a bank of	
	three WaveTek #753A "brick-wall" filters, and three KronHite	
	#3343 filters set to low-pass mode. Finally, the Ithaco pre-	
	amplifiers are located at the bottom of the rack. The computer	
	on the right of the equipment rack was used to remotely con-	
	trol the recording tape deck and digitize and store the outputs	
	from the precision electronic mass flow meter and the pressure	
	transducers mounted in the bubble maker	149
5.9	Incident main bang (a) and backscattered (b) waveforms from	
	the steel target sphere measured at receiver H1. Here, the driv-	•
	ing frequency is 500 Hz and the backscatter return is received	
	at $\triangle t_{\rm meas} = 14.5  {\rm ms}$ , relative to the incident pulse. To enhance	
	the echo, an additional 20 dB of gain was added to (b)	152
5.10	Time gated echo response from a rising bubble cloud measured	
	by receiver H1. Here, the PS was driven at 3000 Hz. The time	
	duration of this oscilloscope trace is 50 s. For each incident	
	main bang pulse, a 10 ms window centered over the expected	
	backscatter signal position (between 15.5 and 25.5 ms) after the	
	leading edge of an incident main bang pulse) was digitized and	
	stored to memory. The pulse repetition frequency was 2 Hz and	
	thus 100 such echoes are displayed in a compressed form (10	
	echoes per division). The cloud was released within $\pm 1\mathrm{s}$ of	
	the initial trace and the peak echo response occurs between the	156
	10–15 s mark	156

5.11	Backscatter target strength measured below 1600 Hz. The hol-	
	low squares were obtained using the parametric source; the solid	
	squares using the conventional source. The error bars indicate	
	the extent of deviation in the backscatter target strength over	
	a consecutive series of three bubble cloud releases. Note: The	
	curve joining the points is only used to train the eye	157
5.12	The data tapes were mounted on the reel-to-reel tape deck and	
	digitized. Prior to digitizing, an anti-aliasing filter set to one-	
	half the data acquisition rate was placed between the tape deck	
	output and the digitizer input. The computer (Apple Macin-	
	tosh Quadra 840av) was outfitted with a National Instruments	
	NB-MIO-16 multifunction I/O board and performed the wave-	
	form digitizing. The digitized waveforms were stored to files on	
	magneto-optical disks for future processing	159
5.13	Sample incident and scattered pulse $\approx 11\mathrm{s}$ after cloud release	
	(backscatter position) for a driving frequency of 1400 Hz and	
	measured at H1. To increase the resolution of the scattered	
	echo, the signal was pre–amplified 43 dB prior to recording. The	
	main bang signal was pre–amplified separately by 20 dB. The	
	incident pulse length is 12 ms. The darker region in each plot	
	corresponds to an 8 ms window over which the rms measure-	
	ments were made. In this case, the rms voltages are MBV =	
	$0.6V_{rms},\;EV=0.46V_{rms}.$ Thus for a total transmission loss	
	factor of $TLF = -22.38  \mathrm{dB}$ (recall Table 5.2) and the pream-	
	plifier gain settings listed above, the measured target strength	
	as given by Eq. (A.10) is $TS = -2.9  dB$	162
5.14	Echo response after a cloud release at a driving frequency of	
	1400 Hz	163
5.15	Time elapsed target strength for a driving frequency of 1400 Hz	
	during the Seneca-3 experiment. The crosses, stars, and plus	
	signs are the elapsed time dependent TS measurements for in-	
	dividual cloud releases. The boxed points and the dashed line	
	joining these points indicate the average elapsed time TS. The	
	minimum measurable signal in the absence of a rising bubble	167
- 10	cloud was -26 dB	167
5.16	Summary rise-time dependent TS measurements. Here, the tar-	
	get strength is computed from the average echo voltage return	
٠	from a sequence of three rising bubble clouds. The individual	
	data runs are plotted in Appendix E	168

5.17	Backscatter TS measurements (rms voltage method) performed during the Lake Seneca experiment and comparison with the model. The curves are calculations based on the non-acoustically determined parameters obtained in Chapter 4.3	170
6.1	Near surface scattering geometry. It is important to note that for backscattering $\theta_i = \theta_s \equiv \phi_g$	182
6.2	Low-frequency backscatter from 5 cm radius cloud for $\beta = 5 \times 10^{-3}$ and $\phi_g = 10^{\circ}$ : (a) depth = 2.0 m; (b) frequency = 786 Hz.	186
6.3 6.4	Monopole resonance backscatter from 5 cm radius cloud for $\phi_g = 10^{\circ}$ . The color scale denotes (a): monopole resonance frequency; (b) through (d): TS at resonance	187
	and $\phi_g = 10^\circ$ : (a) free-field calculations; (b) depth = 2.0 m and $U = 7$ m/s	188
A.1	Geometry of a target strength measurement from an arbitrary scatterer. A plane wave $(p_{\text{inc}})$ is incident on the target from the left. A detector located at point $H$ records the scattered signal $(p_s)$ at a distance $r$ and angle $\theta$ from the forward scattering direction. Here, the origin of the scattered signal is the center of the target.	203
A.2	Geometry for quasi-monostatic scattering measurement. A directional source (S) is aimed at the target (T). Both the incident and scattered waveforms are measured at the receiver (H). The distances between the components are known accurately. The incident pressure amplitude measured at the receiver H is $p_{i,H}$ . Incidentally, the source and receiver are separated by a large distance so that the incoming waves appear planar. Similarly, the scattered pressure evaluated at unit distance from the target T is $p_{s,T}$ . H is small enough that it does not interfere with	d'
	the incident field	205

A.3	(a) Incident and scattered signal and (b) same as (a), except	
	added gain to enhance the echo waveform. The signals were ac-	
	quired from a single hydrophone. The source driving frequency	
	is 1400 Hz, and the incident pulse length is 12 ms. The darker re-	
	gion in each plot corresponds to an 8 ms window over which the	
	steady state rms voltage amplitude measurements were made.	
	In this case, the rms voltages are MBV = $0.6  V_{rms}$ , EV = $0.46  V_{rms}$ .	
	The total transmission loss factor is $-22.38 \mathrm{dB}$ . Thus, the mea-	
	sured target strength as given by Eq. (A.10) is -2.9 dB	209
A.4	Fourier transform of incident and scattered waveforms shown	
	in Figure A.3. The driving frequency is 1.4 kHz, and the pulse	
	length is 12 ms. In these figures transform amplitudes are nor-	
	5	213
T2 1	Time alamand TC (dP) various rise time (s)	231
E.1	Time elapsed TS (dB) versus rise time (s)	
E.2	Time elapsed TS (dB) versus rise time (s)	232
E.3	orași	233
E.4		234
E.5	Time elapsed TS (dB) versus rise time (s)	235

### Chapter 1

#### Introduction

Until 1985, the primary source mechanism for the wind dependent ambient noise<sup>1</sup> spectrum after the appearance of breaking waves on the sea surface in the frequency range 20 to 2000 Hz was not well understood. Below 2000 Hz, it has proven difficult to explain the ambient noise spectrum primarily since the localization of the source mechanisms is difficult without the use of a large directional receiver. Carey and Wagstaff (1986) and Carey and Bradly (1985) summarized experimental evidence which strongly suggests that low frequency noise is wind dependent and locally generated. Although it had been suggested for many years that bubbles play a significant role in the creation of ambient noise at the higher frequencies (Wenz, 1962; Kerman, 1984), the problem of ambient noise generation at the lower frequencies was unclear.

Carey and Bradly (1985) and Prosperetti (1985) first suggested that collective oscillations of bubble plumes or clouds near the ocean surface con-

<sup>&</sup>lt;sup>1</sup>An informal definition of ambient sound or ambient noise is the background sound that is unwanted by anyone performing an acoustic measurement.

tributed to low-frequency wind-dependent ambient noise as a natural source mechanism. Collective oscillations is a phenomenon in which individual bubbles in a mixture pulsate in unison at frequencies well below the natural resonance frequencies of the bubbles, resulting in a significant reduction in the sound velocity. Needless to say, this phenomenon and the candidate source mechanisms were met with less than universal enthusiasm. However, in the past decade, many laboratory and field experimental efforts have established that bubble cloud production is a viable source of low frequency sound in the ocean (Roy et al., 1992; Carey et al., 1993; Kolaini et al., 1994; Yoon et al., 1991; Nicholas et al., to be submitted; Hollet, 1994; Farmer & Vagle, 1989).

As early as 1946, it was established that the scattering of sound from ship wakes was highly dependent on the population or density of bubbles entrained by turbulence or formed by propeller cavitation (NDRC, 1946). The fundamental question to be answered is: what role, if any, do bubble clouds play in sea surface scatter? Given that bubble clouds radiate sound at low frequencies, it seems plausible that such clouds might be efficient scatterers of sound as well. This hypothesis was recently tested and verified in a well controlled field experiment in a freshwater lake in New York state. The experimental design, analysis, and associated modeling effort is the topic of this dissertation.

#### 1.1 Ambient noise in the ocean

Prior to World War II, literally nothing was known about oceanic ambient sound in a scientific context. Since that time, a prolific effort devoted to the experimental characterization and theory of ambient noise production has generated hundreds of publications. The first to make an effort at measuring the noise spectrum was Knudsen et al. (1948). Early in the war, his group performed a series of ambient noise measurements in a number of bays, harbors, and near off-shore areas. For obvious reasons, this data was not obtained in the open sea. His results are known as the "Knudsen Curves" and display an increasing source level with wind speed. The slopes of the curves are not wind speed dependent and average -5 dB per octave in the frequency range 2–20 kHz. At the time, the source mechanisms were unknown. He commented that the near-surface noise contributions can be attributed to individual waves and whitecaps, but that these momentary variations are not important at depth (in his case 20 to 300 ft). These curves were regularly used in the years following the war to estimate the noise levels in deep water.

In the years after the war, many experimental efforts were performed in deeper water which confirmed the spectral slope of the Knudsen spectra, but also encompassed a much broader frequency range both above and below Knudsen's experimental capabilities. Following Urick (1984), the noise spectrum is divided into five distinct bands as follows: Ultra–Low Band (< 2 Hz); Infrasonic Band (2 to 20 Hz); Low Sonic Band (20 to 200 Hz); the High Sonic Band (200 Hz to 20 kHz); and the Ultrasonic Band (> 20 kHz). A summary of the noise contributions and the sources encompassing these bands is illus-

trated in Figure 1.1 (Wenz, 1962). It is clear that the noise spectrum slopes primarily in a negative direction, but some regions possess a flat or positive slope. The wind speed dependence in the figure is denoted by the Beaufort wind force presented in Table 1.1.

Beaufort Force	Sea State	Mean Wind Speed (m/s)	Appearance	Probable Wave Height (m)
0	0	0	Sea like a mirror	0
1	1/2	1	Ripples w/out foam crests	0.08
2	1	2.5	Small wavelets; no breaking	0.15
3 .	. 2	4.5	Large wavelets; crests begin to break	0.6
4	3	6.5	Small waves, becoming longer; common breaking	1.1
5	4	9.5	Moderate waves, long form; maybe some spray	1.8
6	5	12.5	Large waves begin to form. The white foam crests are more extensive everywhere	2.9
7	6	15	Sea heaps up and white foam from breaking waves begins to blow in streaks along the direction of the wind.	4.1

Table 1.1: Beaufort scale of wind force and sea state.

The focus of this work is in the frequency range 20 to 2000 Hz for which ambient noise displays a dominant dependence on shipping traffic and wind speed. In remote areas where shipping noise is not prevalent (mostly in the southern hemisphere), the noise spectrum is clearly dominated by wind speed alone at the upper and lower frequencies (Kibblewhite et al., 1976). Furthermore, Whittenborn (1976), Bannister et al. (1981), and Kewley et al. (1990) recognized that there are at least two wind dependent mechanisms responsible for ambient noise; one occurs before the onset of wave breaking, and one after. Kennedy and Goodnow (1990) also reported a wind speed dependence

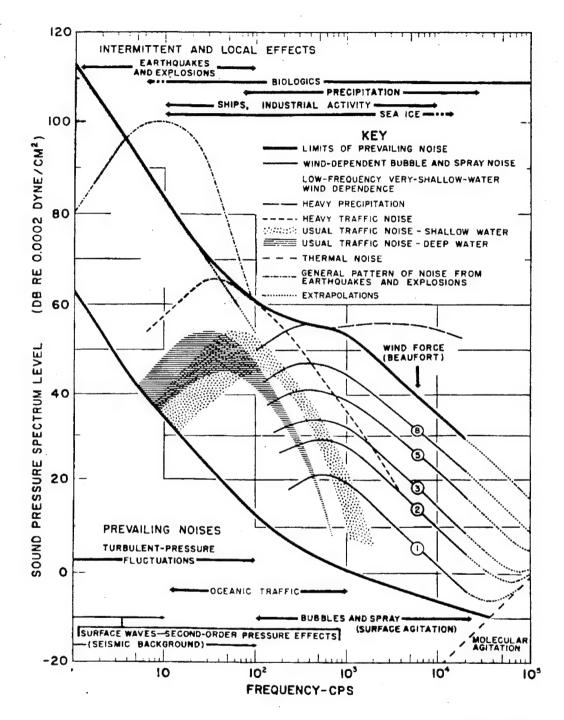


Figure 1.1: Ambient noise spectrum and source mechanisms. From Wenz (1962). The levels in this figure should be increased by 66 dB to conform to the current standard (dB re  $1 \mu Pa$ ).

on the vertical directionality of ambient noise. For wind speeds exceeding the wave breaking threshold, the noise field exhibits a vertical directional spectra consistent with a distribution of dipole sources (40–4000 Hz) near the sea surface.

In Urick's review of ambient noise in the ocean (1984), the wind-dependent processes hypothesized to contribute to surface noise generation after wave breaking includes wind turbulence coupled into surface waves, pressure deviations due to surface motion, nonlinear wave—wave interactions, and spray and cavitation. Furduev (1966) suggested that noise emissions from cavitating bubbles might be a key source near the sea surface. The driving mechanism for cavitation is theorized to be the changing pressure under surface waves which results in bubble growth by 'rectified diffusion' and a subsequent violent collapse. Bubbles entrained by impacting spray from whitecaps are also thought to contribute to the noise spectrum (Wilson, 1980); this view is supported by the experimental data reported by Morris (1978).

Kerman (1984) correctly surmised that densely populated bubble formations created by breaking waves played a significant role in ambient noise production for frequencies above 50 Hz. Furthermore, he speculated that there are two regions of low frequency wind dependence, one preceding to and one following wave breaking, indicating the existence of separate source mechanisms. He recognized that the popular forcing mechanisms of the day (cavitation, spray, large amplitude non linear forcing at frequencies below the bubble resonance, and simple volume pulsations) were inadequately formulated. Nevertheless, he chose to model 'weak cavitation'—the below resonance forcing due

to large amplitude turbulent flow in transient bubble populations—as a candidate mechanism. Recently, however, Prosperetti and Lu (1988) discounted the likelihood that cavitation is a natural source mechanism since the extreme pressure fluctuations required are unrealistic.

For frequencies above  $10\,\mathrm{kHz}$ , the wind dependent noise is due, in part, to single bubble resonant (volume) pulsations driven by the bubble formation process and by turbulent flow<sup>2</sup>. However, in the frequency range 200 to 2000 Hz the oscillation of single bubbles is not thought to significantly contribute to the ambient noise spectrum since such bubbles would have to approach diameters greater than 3 mm. Although bubbles of this size do appear beneath breaking waves, they are extremely buoyant and rise to the surface quickly (Lamarre & Melville, 1992). Furthermore, the bubble distribution is thought by some to peak near  $80\,\mu\,\mathrm{m}$  (Wu, 1981), which would seem to suggest that the role played by single bubbles below 2000 Hz is minimal. During the period following Knudsen's initial measurements during WW–II until Urick's review of ambient noise in the sea in 1984 the structure of the noise spectrum (20 to 2000 Hz) was well documented, but not well understood.

<sup>&</sup>lt;sup>2</sup>The resonance frequency, in Hz, of an air bubble in water at atmospheric pressure is given approximately by  $f_o = 3.2/R_o$ , where  $R_o$  is the bubble radius in m.

# 1.2 The Theory of Low Frequency Ocean Ambient Noise: A Paradigm Shift

The understanding of natural source mechanisms of ambient noise recently became a subject of renewed interest in the ocean acoustics community. This began at the Fall 1985 meeting of the Acoustical Society of America (ASA), at which point the current base of knowledge was reviewed. Since then, no less than 12 sessions devoted to ambient noise processes have been convened at the ASA. In addition, a trio of international symposiums entitled Sea Surface Sound were conducted in 1987, 1990, and 1994. These were multi-disciplinary efforts attended primarily by hydrodynamicists and acousticians who had a common interest in addressing the unresolved issues related to natural source mechanisms of surface generated noise in the ocean. By far, the most controversial topic of discussion at these meetings was the role played by bubbles and the collective oscillations of bubble ensembles in wind dependent ambient noise generation at frequencies below 2000 Hz.

Collective oscillations is not a new idea. In fact, Wood (1941) was among the first to develop a model describing this behavior at frequencies well below the bubble resonance. Over the years, many laboratory studies were conducted which confirm its utility (Silberman, 1957; Fox et al., 1955; Karplus, 1958; Gouse & Brown, 1964; Ruggles, 1987; Lamarre & Melville, 1994; Cheyne et al., 1995). In a two-phase mixture of air bubbles in water, and for frequencies well below individual bubble resonance, the effective mixture

density and compressibility are described by

$$\rho_e = (1 - \beta)\rho + \beta\rho_g, \qquad (1.1a)$$

$$\kappa_e = (1 - \beta)\kappa + \beta\kappa_g, \tag{1.1b}$$

respectively. Here,  $\beta$  is the void fraction or ratio of gas volume to total volume (often expressed as a percentage);  $\rho$ ,  $\kappa$ ,  $\rho_{\rm g}$ , and  $\kappa_{\rm g}$  are the densities and compressibilities of the liquid and gas, respectively. The sound velocity in the mixture is given approximately by

$$c_e \approx (\rho_e \kappa_e)^{-1/2} \approx \sqrt{\gamma P_\infty / \beta (1 - \beta) \rho},$$
 (1.2)

which follows directly from the equation of state (Wood, 1941). Thus, for a mixture having void fraction  $\beta = 10^{-2}$ , the effective sound speed is  $c_e \approx 100 \,\mathrm{m/s}$ ; significantly lower than the speed of sound in the gas! In Chapter 2, the interesting subject of sound propagation in bubbly mixtures will be discussed more formally, including the dispersive characteristics.

The question that remains is: how might an isolated, densely populated bubble-water mixture (i.e., a bubble cloud) contribute to the low frequency noise spectrum? For the purposes of this study a bubble cloud is defined as a mixture of air bubbles and water where  $\beta \approx 10^{-2} \, \text{to} \, 10^{-5}$ , is assumed to be spherical and has a radius  $a < 1 \, \text{m}$ . It is entrained beneath the surface via a breaking wave and persists for a period of only a few seconds (Monahan & Lu, 1990). Is it plausible that such a spherical cloud, characterized by an effective density and sound speed as given by the collective oscillations model,

will radiate sound and resonate at its fundamental eigenfrequency given by the simple approximate expression (Carey & Fitzgerald, 1993; Carey & Roy, 1993; Roy *et al.*, in press)

$$\Omega_0 \approx \frac{1}{2\pi a} \sqrt{\frac{3\gamma P_\infty}{\beta (1-\beta)\rho}} ?$$
(1.3)

For example, a spherical cloud of radius  $a=0.1\,\mathrm{m}$ , possessing void fraction  $\beta=10^{-2}$ , will resonate at approximately 274 Hz. This will be shown explicitly in Chapter 3, and is true regardless of the bubble size distribution, provided that the resonance frequency of the largest bubbles is much greater than the acoustic (driving) frequency.

The acoustic forcing which excites the low frequency modes of a bubble cloud is thought to be individual bubble formation and turbulent flow (Oğuz, 1994). As stated by Longuet–Higgens (1993):

"Each bubble, when it is formed, emits a short acoustical pulse lasting for a few milliseconds only—a kind of birth yell. In listening to underwater sound at these frequencies we are, it appears, hearing a chorus of such birth yells."

The so called 'birth yell' is in fact a damped-resonant volume oscillation. Since no sound of finite duration can be a pure tone, a short duration pulse is composed of a broad spectrum of frequencies which in turn excite the low frequency eigenmodes of the cloud.

## 1.2.1 Experimental evidence of low-frequency noise production by bubble clouds

Several laboratory measurements of the sound produced by bubble clouds have been performed in recent years. Some were conducted using continuously rising bubble columns with the goal of confirming that the collective oscillations model could be used to explain the low–frequency sonic emissions. Others were conceived with the notion that breaking wave occurances are discrete, transient events that entrain large amounts of air, and therefore might explain the low–frequency bursts of sound observed in the ocean.

Yoon et al. (1991) conducted a laboratory experiment in which a cylindrical bubble column was generated by an array of hypodermic needles located at the bottom of a fresh water tank. The peak low frequency acoustic signature from the bubble column (260 to 550 Hz) matched very well the theoretical frequencies predicted by the bubble cloud model of Lu et al. (1990). Similar modeling and measurements by Koller and Shankar (1993; 1994) have shown that the pressure field outside the bubble mixture was, however, evanescent rather than oscillatory.

Nicholas et al. (1994) conducted an experiment with an artificial cylindrical bubble cloud bounded at the top and bottom by pressure release surfaces. In this study, a well controlled, continuously rising cylindrical column of bubbles was produced in a freshwater tank and the higher order radiation modes were found to be in good agreement with the collectively oscillating bubble column model presented in of Lu et al. (1990) and in Commander and Prosperetti (1989). This was followed by an experiment involving bubble

columns created in salt and fresh water (Roy et al., 1991), with special care taken to insure that the column dimension and void fraction was constant in the two experiments. Although the relative bubble size distributions differed significantly, the peaks in the noise spectrum generated by fresh and salt water columns were virtually identical provided the void fraction and column dimensions were held fixed. This confirms the notion that it is the effective sound speed (i.e., void fraction), not bubble size that dictates the low-frequency response of a bubbly assemblage.

Kolaini et al. (1993) measured the low frequency acoustic resonance signature of bubble plumes generated by dropping masses of water from cylindrical containers into a large tank and a pool. They observed that low frequency sound was generated the instant that a large "substructure" detached from the plume. Their observations showed that the substructures were spherical regions of very high void fraction and they demonstrated that the lowest resonance frequency would match the measurements if the mean void fraction was assumed to be approximately 40%.

Kolaini followed the laboratory experiments with a series of well controlled field experiments in which a cylindrical container of water was released into a freshwater lake and the sea water of Puget Sound (Kolaini et al., 1994). Again they observed a damped resonant oscillation as spherical region 'pinched-off' from the cloud. Unlike Roy et al. (1991), they found that the resonance frequency of the newly formed plumes was slightly increased in the marine environment compared to the fresh water of the lake. This, in part, was due to a 10–15% smaller plume radius in the Puget Sound experi-

ment, but also likely due to slight differences in the void fraction, which was not accurately measured. In addition, the salt water emissions were typically 3–4 dB lower, which was attributed to the presence of significantly smaller radius bubbles. The radiation pattern was dipole, which is consistent with the notion that the source is a compact monopole in the vicinity of a pressure release surface.

Finally, recent measurements by Carey et al. (1993) on the sound generated by a tipping trough in both fresh and salt water have shown that the bubble plumes radiate sound at low frequencies in a manner that was consistent with theory and that the resonance frequency was in agreement with a modified Minneart expression (Eq. 1.3) first derived by Carey and Fitzgerald (1993). While the aforementioned experimental measurements were conducted in the near field, this experiment has the distinction of being the first in which the sound radiated by a controlled bubble plume was measured in the far field, with known volume fractions and salinity concentrations. In reverberant tanks, they found that the frequency of oscillation did not change between fresh and salt water. This was determined to be due to the fact that the plume had similar dimensions and void fractions in both cases. It was clear from their measurements that the bubble population was different in fresh and salt water, thus once again it is concluded that the low frequency acoustic characteristics of a bubble cloud are dictated by the void fraction, and not the individual bubble dynamics. The main difference observed between fresh and salt water was in the level of the acoustic radiation. It was found that the salt water case had lower levels due to the higher proportions of small bubbles, resulting in

greater surface area and correspondingly greater attenuation.

Taken as a whole, these studies conclude that reasonably densely populated bubble assemblages radiate sound at frequencies much too low to be described by the resonance of individual bubbles. Furthermore, the theoretical models achieve good agreement with the laboratory studies using a well established collective oscillations model. In such a case, the sound speed at low frequencies is reduced (primarily depending on the void fraction) and the fundamental frequencies of the cloud or plume are determined by the eigenmodes of the system—that is, the resonance frequency is based on the effective sound speed  $(i.e., \beta)$  and length scales of the cloud. At the fundamental resonance frequency, these assemblages possess a characteristic dimension which is less than the acoustic wavelength and can thus be regarded as acoustically compact. In such a case, the source is well approximated in the far field as a monopole source. When placed near a pressure release surface, the radiation from a monopole source takes on a dipole characteristics (i.e., using the method of images). The most important parameter in describing these phenomena is the void fraction. Unfortunately, there is little quantitative data reported on the void fraction of clouds and plumes near the sea surface.

# 1.3 Breaking Waves, Bubble Clouds and Low-Frequency Ambient Noise

In the deep ocean, it is well known that the propagation of surface waves obeys the following dispersion relation (Lighthill, 1975):

$$\Omega^2 = gK(1 + \tau K^2),\tag{1.4}$$

where  $\Omega$  is the radian ocean wave frequency, K is the wave number of the traveling wave, g is the acceleration of gravity, and  $\tau$  is the ratio of the surface tension to the product of g and the water density.

After sufficient periods of high winds, the sea reaches an equilibrium state. The wave height and frequency spectrum can then be predicted by one of several wind–speed based empirical models (Pierson & Moskowitz, 1964; Donelan et al., 1985; Hasselman et al., 1976). Waves of this nature are wind driven, and build up momentum until at some point they break, forming white–caps and entraining air beneath the surface. The onset of this wave breaking process typically begins at wind speeds exceeding  $U > 6 \,\mathrm{m/s}$ .

It is not the instantaneous wind speed, but rather the significant wave height that characterizes the sea state. The Beaufort scale of wind force (Table 1.1) describes the sea state in terms of significant wave height and characteristic appearance (Allen, 1983). The first column lists the Beaufort number, the second the sea state, the third column lists the wind speed required to build up the sea (over a period many hours), and the last column lists the

mean probabilistic wave height. A wind of a given Beaufort force will produce a characteristic appearance of the sea surface provided that it has been blowing for a sufficient length of time, and over a sufficiently long fetch. The term fetch refers to the length of the stretch of water over which the wind acts on the sea surface from the same direction (the longer the fetch, the greater the disturbance for a given wind speed). Small breaking waves (or breakers) begin to appear on the surface of a fully developed sea at the critical windspeed  $(U \approx 6 \,\mathrm{m/s})$ . At this point, the Beaufort wind force is 3 and sea state is 2. As the wind speed increases, the disturbances at the sea surface become more frequent and more violent. For sea states between 3–5 a significant number of white caps appear, followed by sea spray.

According to a model proposed by Monahan and Lu (1990), bubble assemblages created by breaking waves fall into three distinct categories determined by their lifetimes, volume fractions, and penetration depths. The bubble ensembles are known as Type–A, Type–B, and Type– $\gamma$ , and are summarized in Table 1.2 and described below. First, as a wave breaks, a white–cap appears on the surface marking the birth of a Type–A *plume*. Bubbles in this densely populated formation penetrate to depths near 0.25 m, maintaining contact with the surface in an area of approximately 0.5 m<sup>2</sup>. Typical void fractions for these plumes are on the order of a  $10^{-2}$ , but values exceeding this by an order of magnitude have been observed (Lamarre & Melville, 1994; Farmer, 1992).

The Type–A plume quickly decays into a Type–B white–cap in roughly  $0.5\,\mathrm{s}$  and is indicated by the presence of a white foamy patch on the sea surface.

Type	depth (m)	Area (m <sup>2</sup> )	Void Fraction	Life time
A	0.25	0.5	< 0.10	0.5 s
В	0.5	up to 1	$10^{-4} \ 10^{-3}$	4.0 s
$\gamma$	10	10	$O(10^{-6})$	10-100 min

Table 1.2: Characteristics of bubble ensembles following a breaking wave event.

Continued momentum transfer forces the bubbles beneath the whitecap to depths on the order of 0.5 m, as the now detached *cloud* continues to grow in size. In the next 4 s, the void fraction of this densely populated cloud decreases to the order of 10<sup>-4</sup>, occupying on the average a cross sectional area of 1 m<sup>2</sup>. The decrease in void fraction is due to the increase in cloud volume and the loss of larger bubbles rising to the surface.

The larger bubbles in the cloud continuously rise to the surface, leaving behind smaller bubbles which can be stabilized by surfactants (chemical pollutants) and convected to depths on the order of 10 m by a mechanism known as Langmuir circulation<sup>3</sup>. This formation is known as Type $-\gamma$ , or tenuous assemblage, and is noted for void fractions less than  $10^{-6}$  and lifetimes of minutes to hours at a time. Eventually, the 'stabilized' bubbles dissolve into solution.

The distinction between clouds and plumes is clearly demarcated by their respective lifespans, penetration depths, void fractions, and attachment to the surface. That is, plumes are relatively small, densely populated ensembles attached to the surface and lasting only for brief periods of time. While clouds, on the other hand, are detached from the surface, have lifetimes on the order of 4 s, void fractions in the range  $10^{-5}$  to  $10^{-2}$ , volumes less than  $1 \, \text{m}^3$ ,

<sup>&</sup>lt;sup>3</sup>Langmuir circulation is a down welling turbulent current aligned with the wind which convects micro-bubbles to depth (Thorpe, 1984).

and penetration depths on the order of 1 m. The tenuous assemblage has a length scale on the order of 10 m, an extremely low void fraction ( $\beta < 10^{-5}$ ), and can persist for hours at a time. In this work, the primary concern is with the relatively densely populated, short–lived bubble clouds, and *not* the surface–attached plumes or tenuous assemblages.

In the introduction of Lamarre's dissertation (1993) is a compilation of experimental efforts aimed at obtaining the bubble size and spatial distributions in the laboratory and in the sea. From his review, it is clear that the typical bubble size in the ocean is less than  $100\,\mu$  m, and that the distributions obey an exponential depth dependence. While, bubbles larger 1 mm radius have been observed in the ocean (Lamarre & Melville, 1992), buoyancy forces cause them to rise to the surface quickly and thus their contribution to the noise spectrum ( $< 2000\,\mathrm{Hz}$ ) is limited.

As previously noted, the speed of and attenuation of sound are affected in a dispersive manner by the presence of bubbles. This is due to the increased compressibility of the mixture while the density of the mixture remains near that of water. In such a mixture, a phenomenon known as collective oscillations can occur in which the sound speed can be lowered dramatically, even to values below that of air, depending on the void fraction. For example, for a void fraction of  $10^{-5}$ , the speed of sound is reduced to about  $170 \,\mathrm{m/s}$ . Also, increased attenuation due to the presence of bubbles in the mixture is observed.

In Lamarre's (1994) recent research effort, the speed and attenuation of sound were measured beneath breaking waves using a pulse propagation delay technique. The sound speed was estimated by dividing the propagation distance by the transit time of the pulse. The distance between the source and receiver was 1.2 m and the measurements were performed at four depths (0.25 m vertically spaced stations just below the surface). They typically observed decrements in the sound speed on the order of 100 m/s and a maximum deviation of 800 m/s, all attributed to the presence of bubbles. The sound speed decrements were found to decrease with depth in accordance with the bubble population density. Although this effort reported decrements in the sound speed in excess of 800 m/s, it is likely that a closer spacing between the source/receiver pairs would provide a more highly localized result. That is, when the distribution of bubbles is not uniform across the 1.2 m separation (suppose one had a smaller densely populated bubble cloud of dimension 0.25 m) the measurement would likely be biased to sound speeds closer to the liquid and therefore would not yield an accurate result for the maximum sound speed decrement in the bubbly region under a breaker. The difficulty in performing this type of acoustic measurement is apparent since it must be conducted at frequencies well below the bubble resonance frequency, which limits one to relatively distant separations.

In a previous experiment, Lamarre and Mellville (1992) used a conductivity probe to measure the void fraction, not the sound speed, beneath breaking waves in the ocean and the laboratory. In their field experiment the reported void fractions at six depths (equally spaced by  $0.15 \,\mathrm{m}$  beginning at the surface) on the order of  $10^{-2}$ , but at times exceeding  $10^{-1}$ . In the laboratory, surface waves were propagated down the length of a narrow flume and made to break above a single fixed-depth void fraction probe. Thus as the plume

beneath the breaking wave travelled past the sensor a temporal image of the localized void fraction was obtained at a particular depth. The probe depth was varied and the measurement repeated for several more breaking events which allowed them to construct a spatial and temporal image of the void fraction beneath a wave. These images suggested that the laboratory plumes were nominally hemi-cylindrically shaped. When the sensors were deployed on a buoy in the ocean, void fractions in the range  $10^{-2}$  to  $10^{-1}$  were typically observed directly beneath an actively breaking wave—in good agreement with the sound speed decrements they later measured using acoustic techniques.

That the Type-A/B bubble assemblages described above are responsible for the discrete bursts of low frequency (LF) sound has recently been established by Hollet (1994) (see also Carey and Monahan (1990)). In this experiment, an end-fired 32 element vertical array (3 nested subelements of 375, 750, and 1500 Hz) was deployed and pointed toward the surface and a land-based video camera focussed on the same patch of the surface were used to correlate visually observed white-cap breaking events with broad band sound emissions below 1500 Hz. These sound bursts typically persisted for a few seconds and then died out, presumably since no more bubbles were being formed.

Simultaneous acoustic and video measurements of breaking wave events were reported by Farmer and Vagle (Farmer & Vagle, 1989). Using an omnidirectional hydrophone deployed 14 m below the surface, and a sub–surface video camera, they correlated sonic emissions as low as 50 Hz occurring simultaneously with visual observation of bubble cloud production beneath a breaking wave. In order to produce sound at a frequency of 50 Hz, a bubble

radius of 6.4 mm is required. These low frequencies are similar to those observed radiating from the bubble clouds produced by the 'tipping-trough' of Carey et al. (1993) and the bucket drops by Kolaini et al. (1994). A subsequent experiment by Vagle and Farmer (1992) made use of the ambient sound radiated by a bubble cloud to precisely track single breaking wave events in both space and time. Here, they utilized a self-contained freely drifting instrument platform deployed to a depth of 50 m and consisting of six receiving hydrophone elements.

In a recent model proposed by Oğuz (1994), the sound radiated by the formation of bubble plumes beneath the sea surface is discussed. In this study, Oğuz presents an idealized geometry where the sea surface is taken as planar and the bubble plumes are hemispherical. A fundamental feature of his model is that the plumes were allowed to grow with time as the air entrainment takes place, and thus the void fraction evolves as well. The acoustic forcing in the mixture is achieved by broadband emissions from newly 'birthed' bubbles in the vicinity of the perfectly reflecting surface (Pumphrey & Ffowcs-Williams, 1990). This modeling effort results in noise levels that are comparable with the reported field measurements of Hollet (1988; 1994) while using a minimal amount of experimental input (i.e., only the bubble size distribution, mean void fraction, and wind speed need to be known), further evidence that the collective oscillations model is on a strong footing.

Currently, the notion that bubble clouds contribute to the production of low frequency ambient noise is widely accepted. The recent field measurements of Hollet and of Farmer and Vagle, when considered along with the good theoretical agreement of the laboratory results of Carey's tipping–trough and Kolaini's bucket–drop experiments suggest that the bubble clouds generated beneath breaking waves are a fundamental contributing source of natural low frequency noise in the ocean. It is interesting to note that there was no mention of low frequency noise production by bubble clouds in Urick's review of ambient noise in 1984. That this paradigm shift in perception occurred within a ten year span and continues to this day<sup>4</sup> is remarkable and forms a good part of the motivation for the work described in this thesis.

#### 1.4 Sea-surface scatter

A related field which has also gained renewed interest in the last decade is that of low–frequency sea surface scattering and reverberation. Scattering can be regarded as radiation from a 'source' which is driven by an incident wave rather than by a local external generator. The scattering sources near the ocean surface include the surface itself, fish, resonant bubbles, and perhaps bubble assemblages beneath breaking waves. Reverberation is the totality of sound energy scattered from a myriad of inhomogeneities present in the sound beam as well as boundaries along its path. Examples of reverberation include volume and surface reverberation. Volume reverberation is defined as the sound returned by scattering centers in a volume of the sea, whereas surface reverberation is the sound returned by scattering centers at or near the sea

<sup>&</sup>lt;sup>4</sup>The author in the spring of 1995 participated in an experiment (SWELLEX-4) with participants from the U.S. Naval Research Laboratory (NRL) and the Scripps Institute of Oceanography designed to measure the sound produced by breaking waves at frequencies below 400 Hz.

surface. The strength, or efficiency, of these scatterers would be expected to be a function of environmental parameters: wind velocity, sea state, whitecap coverage, temperature, fish populations, etc. The primary distinction between volume and surface reverberation is that surface reverberation is regarded as that part which is dependent on the sea state.

Many fundamental research programs in sea surface scattering and reverberation were carried out in the years since WW–II. One should keep in mind these so called fundamental efforts were not fundamental in the usual scientific context. For the most part, they were designed to accumulate information useful in the fields of anti–submarine or pro–submarine warfare rather than aimed at locating and understanding the physical factors which result in the observed spectrum of sea surface scatter. Regardless, much of the work was both seminal and pioneering (NDRC, 1946).

Among the first efforts at understanding the theory of sea surface scatter were undertaken by Eckart (1953). He considered the scattering of sound from a rough pressure release surface using the Helmholtz-Kirchoff integral (borrowed from the optics community), while neglecting the contribution of point scatterers located near the sea surface (i.e., bubbles and biologics). The quantity  $\mu = \exp(-2kh\sin\phi_g)$ , is the amplitude reflection coefficient of a rough surface; the expression  $2kh\sin\phi_g$  is known as the Rayleigh parameter, where k is the sonic wave number in water, h is the rms wave height for a gaussian distributed sea surface, and  $\phi_g$  is the surface grazing angle. Thus, for acoustic wavelengths greater than the scale of roughness, the scattering strength of the sea surface is predicted to decrease with decreasing frequency.

For short wavelengths, the surface scattering strength is relatively independent of frequency (incoherent scattering dominates). This work was an extension of the problem first discussed by Lord Rayleigh (1877) in which the scattering of sound from a sinusoidal corrugated surface was theoretically investigated.

Urick and Hoover (1956) discussed the backscattering of sound from the sea surface, its measurement, causes, and application to the prediction of reverberation levels. Smooth curves representing their measurements are depicted in Figure 1.2. Their experiments, performed at 60 kHz, indicated that for grazing angles below 30°, the scattering strength was relatively flat (independent of grazing angle), but increased with increasing wind speed. Above about 50°, all wind speed cases showed an increasing scattering strength with grazing angle, and at normal incidence the scattering strength decreased with increasing wind speed. They contend that at low grazing angles and for wind speeds sufficient to cause wave breaking, the flatness of backscattering curves suggests that the roughness of the sea surface alone is not the cause of the scattering. They further speculated that the natural formation of bubbles or bubble layers play a role in the scattering of sound from the near surface layer, but at the time evidence of such a mixture layer had not been established.

Chapman and Harris (1962) used explosive charges to illuminate the sea surface and measured the surface backscatter at grazing angles between 15° to 40° in the frequency range 400 to 6400 Hz. The measurements were made over a 52 hour period in which the wind speed varied from 0 to 15 m/s; the sea state varied from 0 to 6. For wind speeds less than 7.5 m/s they observed general agreement with Eckart's predictions; that is, the backscattering strength was

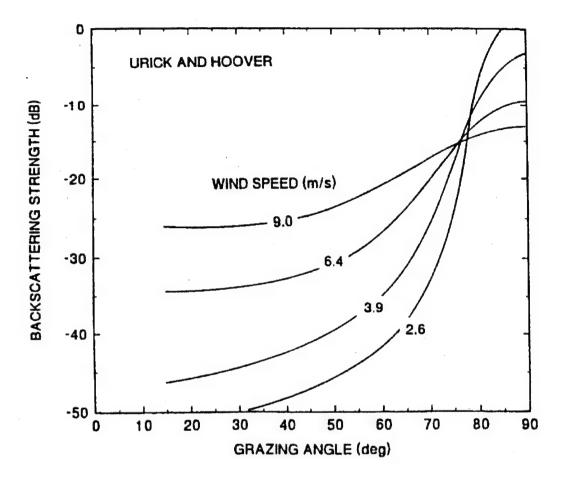


Figure 1.2: Smooth curves of the backscatter from the sea surface as a function of windspeed and grazing angle at 60 kHz (Urick & Hoover, 1956). They speculated that bubbles, or a bubble layer, were responsible for the increase in the scattering for increasing wind speed curves at the lower grazing angles. From McDaniel (1993).

relatively independent of frequency for low wind speeds. However, as the wind speed increased, high backscattering strengths were measured. The data were fitted to an empirical relation based on the Eckart model and referenced to a surface grazing angle of 30°:

$$SS = 3.3\eta \log \frac{\phi_g}{30} - 4.24 \log \eta + 2.6, \tag{1.5}$$

where  $\eta = 158 \left(vf^{\frac{1}{3}}\right)^{-0.58}$ ; v is the wind speed in knots; f is the frequency in Hz; and  $\phi_g$  is the grazing angle. They conjectured that the anomalously high returns were due to a subsurface layer of scatterers, probably of biological origin.

Because of the possible biological contamination, a follow up experiment was performed by Chapman and Scott (1964) in order to measure the sea surface backscatter over an extended range of frequencies (100 to 6400 Hz) and grazing angles ( $\phi_g < 80^{\circ}$ ). In this experiment, the wind speed ranged from 1 to 13 m/s with a mean sea swell of 1.8 m. For the lower wind speeds, the scattering returns were consistent with sea surface scatter—that is, the scattering of acoustic radiation from a surface whose scale of roughness is appreciably greater than the wavelength of the incident sound. At the lower wind speeds (*i.e.*, smaller significant wave heights), the data was not found to be dependent on frequency, consistent with Eckart's surface scattering model. Again, however, the higher wind speed cases showed general disagreement with the surface scattering theory.

The results predicted by Eq. (1.5) cannot be explained by the conventional perturbation or composite roughness models for surface backscatter, especially at shallow grazing angles and low frequencies where the measured scattering strengths are 10 to 100 times stronger (McDaniel, 1993), and it is not likely that advances in the rough surface scattering theories will bridge the gap. Evidently, an unaccounted source mechanism for sea surface backscatter exists, and it appears that this mechanism is present only for wind speeds high enough to cause waves to break. Thus, bubble clouds, layers, and plumes are

likely candidates.

# 1.5 Sea surface scattering: selected field experiments

In the late 1980s and early 1990s a series of tests of low frequency sonar experiments were performed (Critical Sea Tests, CST) in which one of the objectives was to measure boundary scattering strengths at frequencies below 1000 Hz and to investigate the dependence on various acoustical and environmental properties. One of the motivations to perform these tests was to address the observations made by Chapman and Harris (1962), hereafter CH. Ogden and Erskine (1994), like CH, used SUS (explosive) charge sources to insonify the sea surface during periods in which the sea state varied from 0 to 4.5. They performed direct path surface scattering measurements and found general disagreement with the CH empirical model for grazing angles between 10° to 30° and for wind speeds varying between 1.5 and 14 m/s at a center frequency of 70 Hz. However, for a frequency of 930 Hz, a wind speed dependence similar to the CH model was observed—particularly for the higher wind speeds.

Their results, illustrated in Figure 1.3, can be summarized as follows: For relatively calm seas at all frequencies and for rougher seas at the lower frequencies, the scattering strengths are reasonably well characterized by perturbation theory in which air—sea interface scattering is the dominant mechanism (Thorsos, 1990). In the transition region, in which the whitecap coverage of the sea changes from a few to commonly visible, the scattering levels are

difficult to predict with any accuracy. Presumably, a competition between interface scattering and the onset of scattering from bubble assemblages occurs. At higher wind speeds and higher frequencies, the CH empirical formula is found to be consistent with the field data. In this region, the frequency/wind speed/grazing angle dependence given by Eq. (1.5) and calculated using perturbation methods are radically different, strongly suggesting that some other mechanism plays a role.

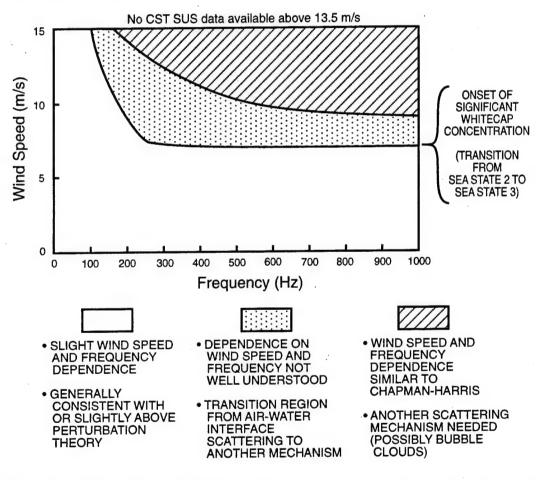


Figure 1.3: Summary of SUS surface scatter results. From Ogden and Erskine (1994).

In the Gulf of Alaska, Gauss et al. (1993) measured surface scattering for a range of significant wave heights (1.5 to 5 m) and very low grazing angles (between 1.5 to 3.5°) using a high resolution receiving array limited to frequencies below 1000 Hz. A vertical array consisting of 10 low frequency and 10 high frequency sources designed to emit 80 Hz bandwidth tones centered at 250 Hz, and 170 Hz bandwidth tones centered at 915 Hz, respectively was deployed at a depth of 150 m. They reported strong evidence of volume scattering (believed to be salmon) in the lower frequency band, and surface scattering for the upper frequencies. However, in the upper band, they also observed prominent, temporally discrete backscatter returns from the surface lasting less than 45 s (the scan repetition rate). The spatial character of these signals revealed strong, isolated features less than 10 m in size. Given the sea state, these transient scattering results are consistent with the spatial and temporal properties of newly formed bubble clouds such as those observed by Farmer and Vagle (1989) and Lamarre and Mellville (1992), and with the notion that such clouds might contribute to the scattering.

Adair and Huster (1992; 1993) managed to image the scattering strength of the upper 5 m of a large patch ( $\sim 1\,\mathrm{km^2}$ ) of the ocean surface using a long towed horizontal line array and conventional wideband processing techniques in an effort to characterize near–surface reverberation at wind speeds exceeding 7 m/s and for a grazing angles near 10°. Two types of sources were used: SUS charges, which yield high signal–to–noise images with 5 m² spatial resolution; and conventional projectors with higher repetition rates (6.7 and 20.1 s). Conventional sources were employed to achieve intermediate range resolution

on the order of 50 m<sup>2</sup>. The SUS experiment should be able to resolve and track the scattering from tenuous bubble *plumes*, which should be extended in space and persist for periods up to tens of minutes.

During periods of no breaking wave activity, they observed isolated bright targets consistent with the scattering strength, frequency response, and migration patterns expected from salmon (see Figure 1.4). Similar bright spots, although more numerous and more broadband, appeared during the high sea state tests. These are believed to be due to scattering from subsurface bubble ensembles since the images showed no ping—to—ping correlation (their lifetimes were less than 10 s). The short lifetimes of the bright echoes are consistent with scattering caused by relatively short lived bubble clouds having void fractions on the order  $10^{-4}$  to  $10^{-3}$  and sizes on the order of 1 m or less. Although features greater than 100 m are shown on the image in Figure 1.4, they reported no long lived, high target strength events.

In a high frequency (3 to 80 kHz) monostatic scattering experiment conducted during wave breaking activity in the North Sea, Nützel and Herwig (1993) and Nützel et al. (1994) observed normal incidence backscatter returning from the sea surface arriving in temporally and spatially discrete 'clumps.' The data was corrected for arrival time differences of the individual echoes due to the varying wave heights, strongly suggesting that bubble clouds were the dominant scattering targets. At 40 kHz, and for a wind speed of 8.5 m/s, the backscatter measurements indicated the presence of transient bubble clouds penetrating to depths of 2 m; at 10 m/s, the clouds penetrated to 5 m; and at 14 m/s, scattering from a continuous layer of bubbles was observed down to

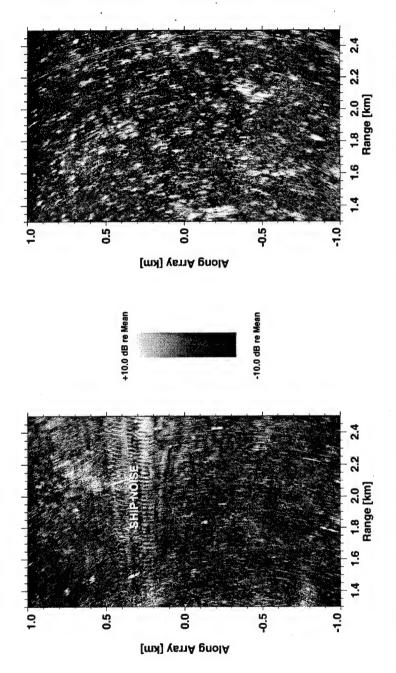


Figure 1.4: Sea surface scattering measurements in the upper 5 m of the ocean surface during CST-7. In the figure to the left, the wind speed was  $U = 4.1 \,\mathrm{m/s}$  (significant wave height 2.2 m, no breaking waves). The average scattering strength in this image is -53 dB. Only a few events, believed to be salmon, exceed the average by +10 dB. In contrast, in the figure on the right, the wind speed is  $U = 16.8 \,\mathrm{m/s}$  (wave height 5.0 m) and a great image resolution is 1 m in range and 0.005 radians in azimuth (120–1430 Hz). This figure was graciously provided number of high target strength scattering events exceeding the average (-30 dB) by +10 dB are prevalent. by Aretè Associates (Miklovic et al., 1995)

6 m and transient plumes at depths of 10 m. Although the acoustic frequencies used during this experiment are beyond the scope of the present study, the reported observations are, nevertheless, enlightening.

### 1.6 Sea surface scattering: modeling efforts

Prosperetti et al. (1993) and Sarkar and Prosperetti (1993) have modeled sub-kilohertz acoustic backscatter at shallow grazing angles (< 20°) from hemispherical, hemicylindrical, and hemispheroidal bubble plumes attached to a idealized plane sea surface. In their calculations, they obtain quantitative agreement with the CH curves using a wind speed dependent white-cap coverage ratio to populate the surface with a density of plumes. The plumes they examined had a radial footprint on the sea surface of 0.5 m, and void fraction in the range 10<sup>-3</sup> to 10<sup>-1</sup>; in the hemispheroidal case the extent of the cloud depth was made to vary by 1-5 radii. Both of these studies were based on a relatively simple acoustic propagation model for a bubbly liquid; a model that has been found to be in excellent agreement with the available data (Commander & Prosperetti, 1989; Lu et al., 1990; Yoon et al., 1991; Silberman, 1957; Cheyne et al., 1995).

In Henyey's (1991) recent study, the well known Born approximation (a weak scattering theory) was used to estimate the acoustic backscatter from the larger tenuous bubble plumes ( $\beta < 10^{-5}$ , depths and lengthscales on the order of 10 m, and lifetimes of tens of minutes to hours) beneath the sea surface. In another study by McDonald (1991), vertically oriented low–void fraction

bubble plume cylinders with rigid boundary conditions were modeled. While both of these efforts were able to show general agreement with the CH empirical equation, they failed to obtain the precise grazing angle dependence obtained by CH. Furthermore, it is not clear, given the available evidence, that any of the models which demonstrate agreement with the Chapman–Harris curves are accurate.

Many investigators currently believe that the global features of the frequency and grazing angle dependence of sea surface backscatter are more closely influenced by the tenuous (low-void fraction) clouds, which occupy a greater fraction of the surface for longer periods of time. This position is not disputed here, for it is to a large extent based on the result of assumptions regarding the statistics and anatomy of oceanic bubble clouds. However, observations of long term, isolated scatterers have not been measured acoustically as would be expected from a tenuous bubble distribution. It seems clear that the high-sea state surface scattering observations that are correlated with spatially and temporally discrete high target strength scatterers (*i.e.*, "spiky returns") are best modeled using short-lived, near surface bubble clouds. Such clouds might possess length scales ranging from 0.1 to 2 m, and void fractions ranging from  $10^{-5}$  to  $10^{-2}$ . They would exist on the order of 10 seconds at depths ranging from just below the surface to 8 m. Is it possible for such small, shallow clouds to generate sub-kilohertz target strengths<sup>5</sup> in excess of -5 dB?

<sup>&</sup>lt;sup>5</sup>By definition, a 1 m-diameter perfectly reflecting sphere has a target strength of -12 dB.

### 1.7 Resonance scattering from bubble clouds

Substantial laboratory and field evidence suggests that acoustically compact (ka < 1), high void fraction bubble clouds oscillate collectively. When excited, these clouds radiate low–frequency sound that possesses spectral features dictated by the normal modes of the cloud; they scale with the cloud dimensions and the mixture sound velocity. It therefore seems reasonable to assume that such a cloud would scatter sound in a similar fashion. This notion was vigorously promoted in the works of Carey et al. (1985; 1988; 1993) and Prosperetti et al. (1985; 1988b; 1988a; 1990; 1993). However, until recently this view had not been experimentally verified for sub–kilohertz frequencies. This dissertation discusses an experiment, and the associated modeling effort, which demonstrates that acoustically compact bubble clouds are effective scatters of sound.

The so called "Lake Seneca Experiment" took place in the summers of 1989 and 1990 at the NUWC fresh-water sonar test facility at Lake Seneca, NY. It was a multi-institutional effort, with contributions from researchers at NUWC, NCPA, BB&N, and the University of Connecticut<sup>6</sup>. The purpose of this experiment was to measure the frequency-dependent backscatter from a well-characterized bubble cloud in the absence of boundaries and for known propagation conditions. This experiment was not designed to duplicate the oceanic environment. Rather the effort was designed to obtain data to test and validate the collective oscillation model hypothesis. The bubble cloud

<sup>&</sup>lt;sup>6</sup>NUWC = Naval Underwater Warfare Center; NCPA = the National Center for Physical Acoustics at the University of Mississippi; and BB&N = Bolt Berenek and Newman

was generated using a transiently-vented compressed air source submerged to 91 m. The rising bubble cloud was insonified using a highly-directional parametric array sound source. Although data was obtained for frequencies ranging from 250 Hz to over 10 kHz, the interest in this effort is primarily in the scattering response of bubble clouds at sub-kilohertz frequencies, where low-frequency sonar systems operate, and much of the existing field data has been obtained. The estimated cloud *in situ* properties were known accurately enough to effectively model the scattering response for frequencies near the lowest-order (monopole) resonance. At these low frequencies, the cloud is acoustically compact, and the details of its shape are unimportant. In essence, the cloud behaves like a volume scatterer and the resonance response is that of a monopole or equivalent pulsating sphere. In this limit, the monopole resonance frequency and target strength are described by simple analytical expressions that effectively highlight the underlying physics and facilitate experimental confirmation.

For the sake of clarity, this dissertation is subdivided into the three parts. Part–I offers a development of the collective oscillations model and the scattering from a high volume fraction bubble cloud in the low–frequency limit, where the resonance of the individual bubbles are much higher than the driving frequency and the effective medium approximation is valid. Following this development, which is derived from the classical literature, theoretical expressions for the frequency–dependent acoustic backscatter from a spherical bubble cloud in the free field are presented. It is important to note that Part I of this dissertation is devoted solely to the modeling of bubble cloud scattering

in the free field. In Part II, the Lake Seneca bubble cloud scattering experiment and associated analysis is described and compared with the model. Finally in Part III, a model of the low–frequency monopole scattering from a bubble cloud near the sea surface is presented. Using the method of images, the predicted low–frequency backscatter target strength is computed as a function of grazing angle, frequency, and wind speed. Once again, the low–frequency, acoustically compact (monopole) scatterer assumption results in expression for which the underlying physics becomes evident. Trends in the calculations are described and comparison is made with field measurements of Adair and Huster (1992; 1993).

Several tacit assumptions will be made in the development of the models herein. First, the bubble cloud is spherical and consists of a mean void fraction mixture. That is, the boundaries of the cloud are well defined, and the distribution of bubbles within the sphere is uniform. Furthermore, the cloud is embedded in a fluid free of other inhomogeneities (*i.e.*, the presence of a bubble layer or other clouds will not be considered). The bubbles, and the cloud, are fixed both temporally and spatially. Although these assumptions are nonphysical from an oceanographic standpoint, the goal here is to consider the scattering response from a single cloud beneath the sea surface, and thus the 'ideal' case will be entertained.

The goal of this dissertation is not to resolve the discrepancies between the sea surface scattering models and the field measurements. Rather, the interest is in the physics of scattering of sound from an acoustically compact bubble cloud both in the proximity and in the absence of the sea surface. The contribution due to scattering from an ensemble of bubble clouds will not be considered. It is a fact that bubble clouds and plumes are produced at the sea surface during wave breaking activity. It is also well established that the generation of these clouds releases a significant amount of acoustic energy into the ocean concentrated in the frequency range 20 to 2000 Hz. Is it plausible then that the presence of these clouds near the sea surface contributes to the overall backscatter observed during wave breaking?

## Part I

Acoustic scattering from bubble clouds in the absence of boundaries: Theory

### Chapter 2

# Propagation of sound in a bubbly mixture

### 2.1 Introduction

The propagation of acoustic waves in fluids is highly influenced by properties such as absorption and scattering. In fact, for a two-phase fluid consisting of water mixed with a sufficiently dense concentration of air bubbles, the sound velocity is frequency dependent. This dependence is a consequence of the marked change of the fluid's natural compressibility when mixed with bubbles (Brekhovskikh & Lysanov, 1991). It will be shown that for frequencies below the fundamental resonance of the individual bubbles, the sound speed can be lowered dramatically. This dependence is solely on the fractional volume or void fraction of bubbles in the fluid (Wood, 1941). The physics behind the lowering of the sound speed is that the bubbly region possesses a density

close to that of the liquid, but a much greater compressibility. The free gas establishes the compressibility, while the water provides inertia. If one recalls that the sound speed in a fluid medium is proportional to the square—root of the inverse product of these quantities, the physical basis for the sound speed defect becomes obvious.

Two models will be considered. First, in the low frequency quasi-static limit a model for the sound propagation in a bubbly fluid similar to that originally proposed by Wood (1941) will be presented. He derived the mixture sound speed based on the assumption that bubbles pulsate isothermally; a good approximation for extremely small bubbles driven at low frequencies. In addition, Wood neglected attenuation.

A second, dispersive model will be presented which spans a much greater frequency range (above and below the bubble resonance frequency) and accounts for energy losses through viscous, thermal, and acoustic damping (Commander & Prosperetti, 1989; Lu et al., 1990). The basis of the dispersion model can be traced back to Spitzer's early work outlined in the classic text *Physics of Sound in the Sea* (1943). In his study, the bubbles oscillations were considered in the adiabatic limit. In the present work, the methodology presented in Commander and Prosperetti (1989) will be followed. The bubble dynamics are treated rigorously so that both the damping 'constant' and the polytropic exponent of the gas in the interior of the bubble are frequency dependent. In both cases the effective (i.e., mixture) properties of the medium will be denoted by the subscript e.

# 2.2 Sound speed in bubbly liquids — Low Frequencies

In the quasi-static limit, the effective medium approximation is obtained by applying an 'order of magnitude' analysis. Let  $\rho$  and  $\rho_{\rm g}$  be the liquid and gas densities in a fixed suspension of gas bubbles in liquid; by suspension, it is meant that there is no relative motion between bubbles and liquid. The bubbles are assumed small and sparsely distributed, but sufficiently densely populated on some macroscopic scale that they can be regarded as determining the properties of a single phase continuum of density  $\rho_{\rm e}$ .

Let  $\beta$  be the void fraction (or concentration) of bubbles (*i.e.*, the ratio of the total gas volume to the total mixture volume). The effective density is defined by

$$\rho_e \equiv (1 - \beta)\rho + \beta\rho_g. \tag{2.1}$$

Because of the assumption of no relative motion, the mass of the gas in a unit mass of mixture must be constant. The volume of unit mass of mixture is  $1/\rho_e$ , the fraction of this volume occupied by bubbles is  $\beta/\rho_e$ , and the mass of gas in this fraction is  $\beta\rho_g/\rho_e$  (Crighton *et al.*, 1992). Hence,

$$\frac{\beta \rho_{\rm g}}{\rho_{\rm e}} \equiv {\rm Constant.}$$
 (2.2)

Furthermore, suppose a *quasi*-uniform pressure exists everywhere within the mixture. This assumption is valid only at very low frequencies—frequencies

far below the bubbles' resonance frequency,  $\omega_0$ , as will be shown in Section 2.3. As usual, the inverse square of the sound speed is given by the derivative of the density with respect to pressure (Pierce, 1989),

$$c_{\epsilon}^{-2} = (1-\beta)\frac{d\rho}{dp} + \beta\frac{d\rho_{\rm g}}{dp} + (\rho_{\rm g} - \rho)\frac{d\beta}{dp}$$
 (2.3)

$$= \frac{1-\beta}{c^2} + \frac{\beta}{c_{\rm g}^2} + (\rho_{\rm g} - \rho) \frac{d\beta}{dp}. \tag{2.4}$$

Now consider the following expression:

$$\beta \rho_{\rm g} = \left(\frac{\beta \rho_{\rm g}}{\rho_{\rm e}}\right) \rho_{\rm e},\tag{2.5}$$

which, upon taking the derivative with respect to pressure and using the definition given by Eq. (2.2) yields,

$$\rho_{\rm g} \frac{d\beta}{dp} + \frac{\beta}{c_{\rm g}^2} = \left(\frac{\beta \rho_{\rm g}}{\rho_{\rm e}}\right) \frac{d\rho_{\rm e}}{dp},\tag{2.6}$$

and finally,

$$\frac{d\beta}{dp} = \left(\frac{\beta}{(1-\beta)\rho + \beta\rho_{\rm g}}\right) \frac{d\rho_{\rm e}}{dp} - \frac{\beta}{\rho_{\rm g}c_{\rm g}^2}.$$
 (2.7)

Substitution of the preceding equation into Eq. (2.4) gives, after some algebraic manipulation,

$$\frac{1}{c_e^2} = \frac{d\rho_e}{dp} = \frac{(1-\beta)^2}{c^2} + \frac{\beta^2}{c_g^2} + \beta(1-\beta) \left\{ \frac{\rho^2 c^2 + \rho_g^2 c_g^2}{\rho \rho_g c^2 c_g^2} \right\}.$$
 (2.8)

For the case where  $\beta$  is not too large, or too small, the first two terms

in Eq. (2.8) can be neglected, and the final term is approximated yielding the effective sound speed

$$c_e^2 \approx \frac{\rho_{\rm g} c_{\rm g}^2}{\beta (1 - \beta) \rho},\tag{2.9}$$

since  $\rho_{\rm g} c_{\rm g}^2 \ll \rho c^2$ .

The sound speed in the gas is  $c_{\rm g}^2 = \kappa P_{\infty}/\rho_{\rm g}$  (Kinsler *et al.*, 1982), where  $\kappa$  is the polytropic exponent of the gas, and  $P_{\infty}$  is the ambient static pressure in the fluid far from the mixture. For isothermal conditions,  $\kappa = 1$  and for adiabatic conditions  $\kappa = \gamma$ , where  $\gamma$  is the ratio of specific heats. A more rigorous model is presented in the next section which describes the the frequency dependence of the polytropic exponent explicitly. Substitute the gaseous sound speed into Eq. (2.9) to arrive at the well known Wood expression (1941) for the speed of sound in a bubbly mixture in the isothermal and adiabatic limits, respectively:

$$c_e \approx \sqrt{\frac{P_{\infty}}{\beta(1-\beta)\rho}},$$
 (2.10a)

$$c_e \approx \sqrt{\frac{\gamma P_{\infty}}{\beta (1-\beta)\rho}}.$$
 (2.10b)

It will be shown in Section 2.3, that these expressions are valid only for frequencies well below the bubble resonance frequency. It should also be noted that for extremely small  $\beta$ , this approximation exceeds the sound speed in the host liquid—a limitation not imposed by Eq. (2.8). For example, in seawater the Wood's equation is valid for  $\beta > 3 \times 10^{-4}$ ; for smaller values it rapidly

exceeds the liquid sound speed.

Figure 2.1 clearly illustrates one of the most remarkable features of sound propagation through bubbly fluids. It is clear that the sound speed in such a mixture of water and air can be much lower than either of the constituents over a large range of  $\beta$ —as low as 20 m/s. The calculation is shown for both isothermal ( $\kappa = 1$ ) and adiabatic ( $\kappa = 1.4$ ) bubble behavior using Eq. (2.10a) and Eq. (2.10b). The experimental data is from Karplus (1958) and Gouse and Brown (1964) and obtained at frequencies of 0.5 kHz and 1.0 kHz. Notice that these measurements correspond to the isothermal model, indicating that the heat transfer from the liquid to the bubbles is enough to maintain constant temperature inside the bubbles.

# 2.3 Sound speed in a bubbly medium — dispersion

The concern here is with the frequency dependence, or dispersion, of sound waves propagating in a two-phase mixture of air bubbles in water. Bubbles are relatively high-Q simple harmonic oscillators and when driven at their fundamental resonance are both strong scatters and absorbers of sound. Actually, sound is attenuated at all frequencies, but at resonance the attenuation reaches a maximum. This indicates that the effective speed of sound has a significant imaginary component which was not addressed in the previous section. Furthermore, the response of the bubble is not uniform over the range of frequencies spanning the bubble resonance frequency. When the acoustic forcing

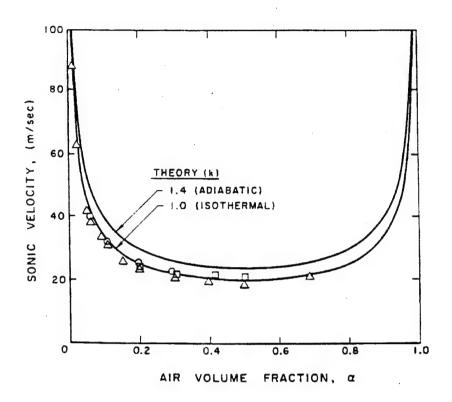


Figure 2.1: Sound speed in a bubbly liquid at atmospheric pressure for  $\kappa = 1.0$  and 1.4 as a function of the void fraction  $\alpha$ . Experimental data from Karplus (1958) and Gouse and Brown (1964) for frequencies of 1 kHz ( $\circ$ ), 0.5 kHz ( $\square$ ), and extrapolated to zero frequency ( $\triangle$ ). From Brennen (1995).

frequency is lower than the resonance frequency, the a bubble will pulsate in phase with the driving force. However, for frequencies exceeding the resonance frequency, the oscillations occur out of phase, and thus the bubble 'appears' stiffer, leading to a corresponding increase in the mixture sound speed.

#### 2.3.1 Theory

In order to understand how harmonic waves of any type propagate, one commonly solves a set of linearized conservation equations which ultimately lead to the Helmholtz or 'wave' equation. To follow in suit, first the conservation equations for a bubbly mixture will be defined. To begin, consider a bubbly mixture as an effective medium with density given by Eq. (2.1). Formally, the void fraction is described by

$$\beta(\mathbf{x},t) = \frac{4\pi}{3} \int_0^\infty R^3(a;\mathbf{x},t) f(a;\mathbf{x}) da. \qquad (2.11)$$

Here, R is the instantaneous bubble radius at position  $\mathbf{x}$  and time t having equilibrium radius a. The quantity  $f(a;\mathbf{x})da$  is the distribution function of the number of bubbles per unit volume with equilibrium radius between a and a + da. Note: for a monodispersed bubble population of radius  $R_o$ ,  $f(a;\mathbf{x}) = \eta \delta(a - R_o)$ , where  $\delta$  is the Dirac delta function, and  $\eta$  is the number of bubbles per unit volume. For simplicity, the spatial dependence of the bubble distribution is assumed to be fixed, but random in spacing. Furthermore, the bubble distribution is assumed monodispersed so that

$$\beta = \frac{4}{3}\pi R_o^3 \eta. \tag{2.12}$$

The averaged continuity and momentum equations applied to bubbly fluids having small  $\beta$  are (Commander & Prosperetti, 1989)

$$\frac{1}{\rho c^2} \frac{\partial P}{\partial t} + \nabla \cdot \mathbf{u} = \frac{\partial \beta}{\partial t}, \qquad (2.13a)$$

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \nabla P = 0, \tag{2.13b}$$

where  $\mathbf{u}$  and P denote the averaged center-of-mass velocity and pressure fields

in the mixture. In order to consider the averaged quantities of the mixture, the nature of the interaction of sound waves with individual bubbles must be determined.

#### 2.3.2 Linearized single-bubble dynamics

Bubble dynamics is a robust field both in terms of theory and experiment. Its origins can be traced to original works by Lord Rayleigh (1877). The individual bubble is well known to be a simple harmonic oscillator for which the time dependent radial oscillations can be described by the Keller equation (Keller & Kolodner, 1956)

$$(1 - \frac{\dot{R}}{c})R\ddot{R} + \frac{3}{2}(1 - \frac{\dot{R}}{3c})\dot{R}^2 = \frac{1}{\rho}(1 + \frac{\dot{R}}{c} + \frac{R}{c}\frac{d}{dt})(p_B - P), \tag{2.14}$$

where R(t) is the instantaneous radius of the bubble; P is the pressure at the bubble location in its absence; and  $p_B$  is the pressure at the bubble interface. Here, the bubble response is assumed to be linear and after making suitable approximations to Eq. (2.14) it can be shown that the radius of, and internal pressure in, the bubble are given by

$$R = R_o(1+X), \quad p = p_e(1-\Phi X),$$
 (2.15)

where  $R_o$  and  $p_e$  are the equilibrium values (Prosperetti et al., 1988).

$$p_{\rm e} = P_{\infty} + \frac{2\sigma}{R_o} \tag{2.16}$$

where  $\sigma$  is the surface tension, and

$$\Phi = \frac{3\gamma}{1 - 3(\gamma - 1)i\chi \{(i/\chi)^{1/2} \coth[(i/\chi)^{1/2}] - 1\}}.$$
 (2.17)

The preceding equation arises from a solution of the thermomechanical behavior of spherical gas bubbles with the additional assumption of small amplitude oscillations (Prosperetti *et al.*, 1988), with  $\gamma$  denoting the ratio of specific heats in the gas. The thermal penetration length is given by

$$\chi = \frac{D}{\omega R_o^2},\tag{2.18}$$

in which D is the thermal diffusivity of the gas (Lu et al., 1990; Commander & Prosperetti, 1989; Prosperetti, 1991). It is possible to show that the response of the bubbles to the harmonically oscillating averaged pressure field is given by

$$X = -\frac{(P - P_{\infty})/\rho R_o^2}{\omega_o^2 - \omega^2 + 2ib\omega},$$
(2.19)

where  $\omega_0$  is the natural frequency of the bubble, and b is the effective damping 'constant' given by

$$\omega_0^2 = \frac{p_e}{\rho R_o^2} \left( 3\kappa - \frac{2\sigma}{R_o p_e} \right) \tag{2.20a}$$

$$b = \frac{2\mu}{\rho R_o^2} + \frac{p_e}{2\rho\omega R_o^2} \Im(\Phi) + \frac{\omega^2 R_o}{2c}.$$
 (2.20b)

Here,  $\mu$  is the liquid viscosity, and  $\Im(\Phi)$  returns the imaginary portion of  $\Phi$ .

Neglecting the effects of surface tension in Eq. (2.20a), the well known Minneart (angular) resonance frequency is readily obtained for the bubble (Minneart, 1933)

$$\omega_0 \cong \frac{1}{R_o} \left( \frac{3\kappa P_\infty}{\rho} \right)^{\frac{1}{2}}. \tag{2.21}$$

Thus the resonance frequency is proportional to the inverse of the radius and in water at atmospheric pressure (i.e., near the surface) is well approximated by the simple expression  $f_o \approx 3.2/R_o$  in Hz.

In Eq. (2.20b) the damping term, b, is a frequency dependent complex quantity which accounts for viscous, thermodynamic, and acoustic losses;  $\kappa$  again is the frequency dependent polytropic exponent (Prosperetti, 1991),

$$\kappa = \Re(\Phi)/3. \tag{2.22}$$

A graph of its values for a bubble of radius  $R_o = 500 \,\mu$  m is shown in Figure 2.2. It is interesting to note that the polytropic exponent approaches unity (i.e., the isothermal limit) well below the bubble resonance frequency ( $f_0 \approx 6.4 \,\mathrm{kHz}$ ), and its adiabatic value of  $\gamma = 1.4$ , the ratio of specific heats for a diatomic gas, at higher frequencies. It seems that only in the extreme limits does the gas in the bubble portray purely isothermal or adiabatic thermodynamical behavior. Table 2.1 lists values of the physical constants and environmental parameters used in the calculations presented in this section.

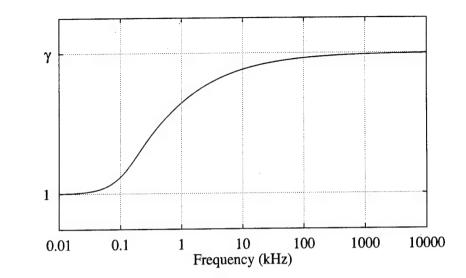


Figure 2.2: Polytropic index versus frequency for a 500  $\mu$  m radius bubble.

### 2.3.3 Effective wave equation

The effective wave equation in the bubbly fluid is obtained by eliminating the velocity field from Eq. (2.13a) and (2.13b), using Eq. (2.12) and (2.19). After linearizing (d'Agostino & Brennen, 1983; Commander & Prosperetti, 1989), the wave equation takes the following form:

$$\nabla^2 P + k_e^2 (P - P_{\infty}) = 0, \qquad (2.23)$$

$\rho$	1024	$kg/m^3$	c	1500	m/s
$ ho_{g}$	1.2	$kg/m^3$	$c_{ m g}$	330	m/s
$P_{\infty}$	$10^{5}$	Pa	T	18	$^{\circ}\mathrm{C}$
$\gamma$	1.4		σ	7.2	N/m
$\mu$	0.001	$Pa \cdot s$			

Table 2.1: Physical parameters used in this section.

where  $k_e = \omega/c_e$  is the effective wave number in bubbly mixture, and  $c_e$  is the effective sound velocity. The dispersion relation obtained is

$$k_e^2 = \frac{\omega^2}{c^2} + \frac{4\pi\eta R_o \omega^2}{\omega_o^2 - \omega^2 + 2ib\omega}.$$
 (2.24)

This wave equation (Eq. 2.23) describes the interesting phenomena alluded to earlier. Clearly the effective wave number in the mixture is frequency dependent and complex. Thus one can expect a maximum in the attenuation at the bubble resonance frequency. For frequencies below bubble resonance, the  $\omega_0$  term in the denominator of Eq. (2.24) dominates yielding a wave number proportional to the driving frequency, and hence an effective sound speed relatively independent of frequency.

The ratio of the sound speed in the liquid to the speed of sound in the mixture is obtained by dividing Eq. (2.24) by  $k^2 = \omega^2/c^2$  yielding,

$$\frac{c^2}{c_e^2} = 1 + \frac{4\pi c^2 \eta R_o}{\omega_0^2 - \omega^2 + 2ib\omega}.$$
 (2.25)

Then for frequencies well below the bubble resonance frequency, the mixture sound velocity can be approximated by

$$c_e^{-2} \approx \frac{4\pi \eta R_o}{\omega_o^2} = \frac{3\beta}{\omega_o^2 R_o^2},$$
 (2.26)

neglecting the damping term. Substitution of Eq. (2.12) and (2.21) into the preceding approximation yields  $c_e = \sqrt{\kappa P_{\infty}/\beta\rho}$ , which resembles our previous approximation Eq. (2.9) to within a factor of  $(1-\beta)^{-1/2}$ , reinforcing our earlier

assumption that below the bubble resonance frequency the mixture sound speed in dependent only on the void fraction.

Let the ratio of liquid sound speed to effective sound speed in the mixture be  $c/c_e = u + iv$ . Then the phase velocity in the mixture is V = c/urather than  $\Re(c_e)$ , because the time and spatial dependence of the harmonic waves in Eq. (2.23) are described by

$$\exp(ik_e x - i\omega t) = \exp(-\omega \frac{v}{c}x) \exp(i\omega(\frac{u}{c}x - t)). \tag{2.27}$$

In the low frequency limit, the square of the phase velocity V is essentially equivalent to the previously derived expression (Eq. 2.9) and (2.26). The fact that the effective medium expressions,  $k_e$  and  $c_e$ , are complex indicates that sound propagation in the mixture is exponentially attenuated. The attenuation coefficient,  $\mathcal{A} = 20(\log_{10} e)(\omega v/c)$ , is attributable to the viscous, thermal, and acoustic energy loss mechanisms which affect the bubble as given in Eq. (2.20b).

### 2.3.4 Calculations and experimental results

To perform the calculations, codes based on the equations in (Commander & Prosperetti, 1989) and (Lu et al., 1990) were originally written in Mathematica and tested using the data sets from these papers and compared to the figures in their papers. To improve the performance vastly, the codes were later cast in the C and C++ programming languages and executed on both Sun workstations and Apple Macintosh computers. Special attention was paid to

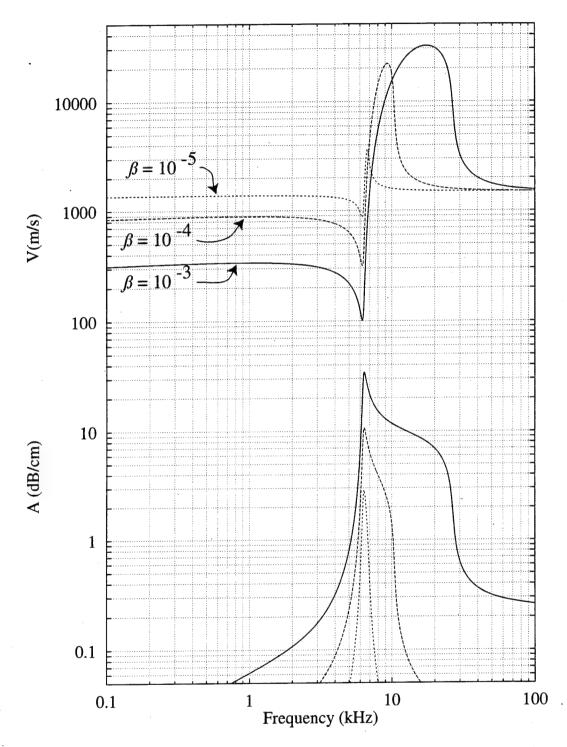
any calculation involving complex arithmetic. To evaluate these expressions  $(e.g., \sqrt(z), z^{-1}, \text{ etc.})$ , standard techniques were used (Abramowitz & Stegun, 1972).

In Figure 2.3, the frequency dependent phase velocity and attenuation are plotted for a monodispersed bubble distribution of radii  $500 \,\mu$  m, and for  $\beta = 10^{-5}$ ,  $10^{-4}$ , and  $10^{-3}$ . The upper set of curves represent the phase speed in the mixture. For the majority of frequencies below the bubble resonance (i.e., the sharp dip), the sound velocity asymptotes as predicted by the Wood's equation (Eq. 2.9) and is nearly flat. The bubble resonance frequency predicted by the Minneart expression (Eq. 2.21) for a  $500 \,\mu$  m bubble is approximately  $6.4 \,\mathrm{kHz}$ . In Table 2.2, a comparison of the limiting forms for the sound speed is given. This reinforces the statement made earlier that for frequencies well below the bubble resonance the sound speed in the mixture is dependent only on the void fraction, and not the individual bubbles per se.

β	Wood's Eq.	Eq. (2.8)	Eq. (2.26)
$10^{-5}$		$1363\mathrm{m/s}$	$1359\mathrm{m/s}$
$10^{-4}$	988 m/s	$850\mathrm{m/s}$	841 m/s
$10^{-3}$	$313\mathrm{m/s}$	319 m/s	$314\mathrm{m/s}$

**Table 2.2:** Comparison of sound speed as given by Wood's equation (Eq. 2.9), Eq. (2.8) and the dispersive phase speed model (Eq. 2.26) evaluated at 100 Hz.

The 'supersonic' region following the bubble resonance broadens with increasing void fraction and is due to the bubbles pulsating out of phase with the pressure disturbance. This causes the mixture to appear stiffer, and hence the corresponding increase in the mixture sound velocity is observed. Well above the bubble resonance frequency, the phase speed asymptotes to its bub-



**Figure 2.3:** Phase speed (upper) and attenuation (lower) versus frequency for  $\beta=10^{-5},\,10^{-4},\,$  and  $10^{-3},\,$  and bubbles of radius  $R_o=500\,\mu\,\mathrm{m}.$ 

ble free value of c in the liquid. The attenuation is shown in the lower trio of curves, the peak in these curves occurs at the fundamental bubble resonance frequency. It is clear that the attenuation increases as the number density of bubbles increases, as one would expect.

For a given void fraction, increasing or decreasing the bubble radius has the affect of shifting the sharp dip at resonance in the phase speed and the peak in the attenuation curve to the left or right, respectively, as illustrated in Figure 2.4 for  $\beta=10^{-3}$  and bubble radii of 1.1 mm and 0.1 mm.

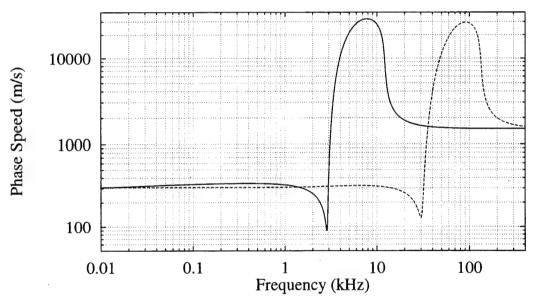


Figure 2.4: Phase speed versus frequency for  $\beta = 10^{-3}$ , and bubbles of radius  $R_o = 1.11 \,\mathrm{mm}$  and  $R_o = 0.1 \,\mathrm{mm}$ .

Silberman (1957) was one of the first to measure the sound speed and attenuation in bubbly mixtures. He filled a rigid cylinder with fresh water and created bubbles by forcing air through hypodermic needles being careful to maintain the water height and void fraction in the tube. Assuming the top and bottom of the tubes were pressure release, standing waves could be

set up in the cylinder well below the tube cut-off frequency. For a given driving frequency, the nodes (or pressure minima) were measured by lowering a hydrophone down the axis of the tube at the following positions  $0, \lambda_e/2, \lambda_e$ , etc., where  $\lambda_e$  is the sonic wavelength in the mixture. Using this information he calculated the mixture sound speed in the tube by multiplying the measured wavelength by the driving frequency. Below the bubble resonance frequency, his measurements were found in good agreement with the models proposed by Wood (Wood, 1941) and Spitzer (1943).

The attenuation due to the mixture was deduced by taking the ratio of the pressure amplitude measurements made at the anti-nodes near the driving transducer (i.e.,  $A = log_{10}(P_1/P_2)/(x_2 - x_1)$ ). Silberman's data compares favorably with Spitzer's classic effective sound speed model outlined in *Physics of Sound in the Sea* (NDRC, 1946). Spitzer's dispersion model assumes that bubbles behave adiabatically, which is reasonable for the relatively large bubbles sizes ( $R_o > 1$  mm) in Silberman's experiment. For smaller bubbles, however, this approach cannot be expected to be accurate, especially since the radial pulsations are closer to isothermal at low frequencies. Also, the advantage of the model outlined in this chapter over Spitzer's is that both the damping constant for the bubble and the polytropic exponent of the gas are dependent on the bubble radius and driving frequency.

Fox et al. (1955) measured the phase velocity in a bubbly mixture by comparing the phase of a received signal in a tank containing pure water and finally a bubbly mixture. Thus, the attenuation measurements were easy to obtain. The agreement with the model is only semi-quantitive due to noise

from standing waves in the tank. This reverberation was likely driven by bubble formation noise.

In comparison with other experimental efforts, Silberman's measurements were unsurpassed in accuracy and control. However, a recent paper by Cheyne et al. (1995) rivals that of Silberman. They adopted the Silberman's standing wave in a tube approach, but developed a non-intrusive measurement technique utilizing a form of Laser Doppler Vibrometer (LDV). Measurements were also performed by lowering a hydrophone into the tube as indicated in the comparison with theory in Figure 2.5. A lightweight shaker was used to drive a piston at frequencies swept from 100 Hz to 20 kHz located at the bottom of the tube. The LDV data were acquired by placing the sensor at the vertical mid-point of the tube and the output recorded using a sweep spectrum analyzer. This resulted in a series of peaks in the frequency spectrum which corresponded to the tube resonances for a given water level and void fraction, and thus the sound speed was easily obtained. Strong attenuation and the extremely low phase speed at the bubble resonance (2.9 kHz) limited the data to values above and below the resonance frequency of the bubbles. Due to the extremely low sound speed at resonance, the tube would have to be less than only a few bubble radii in diameter to satisfy the planewave cutoff criteria. The close agreement with the theory evident in Figure 2.5 was achieved without resorting to fitting parameters. Note, in particular the exceedingly high sound speed measured in the 'supersonic' region around 10 kHz and overall excellent agreement with the model of Commander and Prosperetti.

In Lamarre's (1994) recent research effort, the speed of sound and atten-

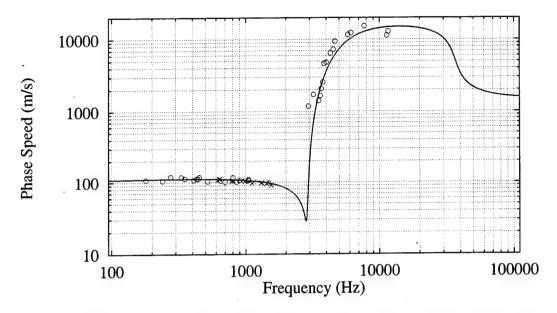


Figure 2.5: Frequency dependent phase speed. The bubble radius was 1.11 mm, and the void fraction  $\beta = 10^{-2}$ . The crosses represent the data measured by lowering a hydrophone down the length of the standing wave tube, and the circles using the LDV. Data courtesy of Cheyne *et al.* (1995)

technique. The sound speed was estimated by dividing the propagation–delay technique. The sound speed was estimated by dividing the propagation distance by the transit time of the pulse. The distance between the source and receiver was 1.2 m and the measurements were performed at four depths (0.25 m vertically spaced stations just below the surface). They typically observed decrements in the sound speed on the order of 100 m/s and a maximum deviation of 800 m/s, which corresponds to void fractions of approximately 0.008 and 0.017 respectively. The sound speed decrements were found to decrease with depth in accordance with the lowered the bubble population. Although this effort reported decrements in the sound speed in excess of 800 m/s, it is likely that a closer spacing between the source/receiver pairs would provide a more highly localized result. That is, when the distribution of bubbles is

not uniform across the 1.2 m separation, say a smaller densely populated region of dimension 0.25 m, the measurement would likely be skewed to sound speeds closer to the liquid and therefore would not yield and accurate result for the minimum sound speed encountered in the bubbly region. The difficulty in performing this type of acoustic measurement is apparent since it must be conducted at frequencies well below the bubble resonance frequency, which limits one to relatively large separations.

In a previous experiment, Lamarre and Mellville (1992) used a conductivity probe to measure the void fraction, rather than the sound speed, beneath breaking waves in the ocean and the laboratory. Each sensor consisted of trio of nichrome wire electrodes spaced 1.6 cm apart and 20 cm in length arranged in a parallel fashion. As bubbles passed between the wires, the conductivity changed depending on the fraction of air occupied by the bubbles. The system was calibrated in a cylindrical tube in which the bubble size distribution, air flow rate, and void fraction were controlled. This method provided a substantially higher localization of measured void fraction, but is limited to larger void fractions in the range  $10^{-3}$  to 0.3—the acoustic technique described above is capable of measuring void fractions down to  $10^{-5}$ . In their field experiment, they reported instantaneous void fractions at six depths (equally spaced by 15 cm, beginning at the surface) on the order of  $10^{-2}$ , but at times exceeding  $10^{-1}$ . In the laboratory, surface waves were propagated down the length of a narrow flume and made to break above a single fixed-depth void fraction probe. Thus as the plume beneath breaking wave travelled past the sensor a temporal image of the localized void fraction was obtained at a particular

depth. The probe depth was varied and the measurement repeated for several more breaking events which allowed them to construct a spatial and temporal image of the void fraction beneath a wave. These images suggested that the laboratory plumes were nominally hemi-cylindrically shaped. When the sensors were deployed on a buoy in the ocean, void fractions in the range  $10^{-2}$  to  $10^{-1}$  were typically observed directly beneath an actively breaking wave—in good agreement with the sound speed decrements they later measured using acoustic techniques.

#### 2.3.5 Bubble size distributions

In order to simplify the analysis, the model presented prior to this point is valid only for a monodispersed bubble radius distribution. However, in the sea, or even in well controlled laboratory experiments, this assumption is likely to break down. The definition of the void fraction (Eq. 2.11) provides a mechanism to include a wide range of bubble radius distributions through the distribution function f(a). The limiting case of a monodispersed distribution is obtained by setting  $f(a) = \eta \delta(a - R_o)$ . A more realistic population would consist of range of radii, not a single radius. Examples might include a discrete bubble radius population for which

$$f(a) = \sum_{m=1}^{M} \eta_m \delta(a - R_{o,m}), \quad m = 1, 2, \dots, M,$$
 (2.28)

where  $\eta_m$  is the 'partial' number of bubbles per unit volume having radius  $R_{o,m}$ . Alternatively, a continuous radius distribution corresponding to the measured populations at sea is given by

$$f(a) = \begin{cases} \eta_o \, a^{-H}; & a_1 < a < a_2, \\ 0; & \text{otherwise,} \end{cases}$$
 (2.29)

where the coefficient H ranges from 3.5 to 5 as indicated in (Lamarre, 1993). Another common continuous distribution is the truncated Gaussian distribution

$$f(a) = \begin{cases} \eta_o e^{-\left(\frac{(a-R_o)}{2\sigma}\right)^2}; & a_1 < a < a_2, \\ 0; & \text{otherwise,} \end{cases}$$
 (2.30)

Here,  $R_o$  is the mean radius;  $\sigma^2$  is the mean square variance about the mean;  $\eta_o$  is the amplitude of the distribution; and the values  $a_1$  and  $a_2$  are the limits of the minimum and maximum bubble radii in the continuous distribution. For any of the above considerations, the integral relation  $(4\pi/3) \int_0^\infty f(a) a^3 da$  must yield the void fraction (Eq. 2.11).

An extension of the collective oscillations model to the more complex bubble distributions is outlined in (Commander & Prosperetti, 1989) and will not be presented in detail here. Instead, the results are given for the effective wave number in the mixture (Eq. 2.24),

$$k_e^2 = \frac{\omega^2}{c^2} + 4\pi \int_0^\infty \frac{f(a)a\omega^2}{\omega_0^2 - \omega^2 + 2ib\omega} da,$$
 (2.31)

and thus the sound speed ratio squared for a multi-dispersed radial distribu-

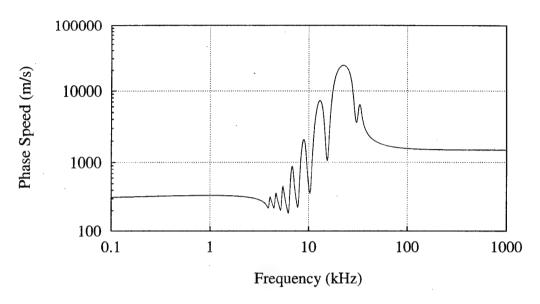
tion of bubbles is

$$\left(\frac{c}{c_e}\right)^2 = 1 + 4\pi c^2 \int_0^\infty \frac{f(a)a}{\omega_0^2 - \omega^2 + 2ib\omega} da. \tag{2.32}$$

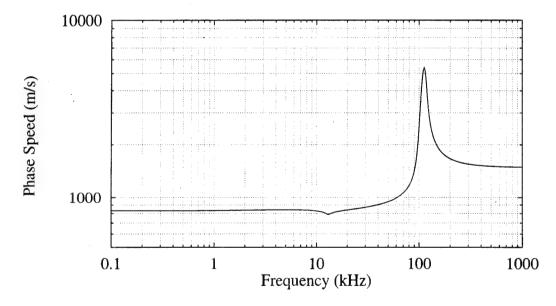
Generally, the distribution functions are truncated over some range of bubble radii as indicated in Eq. (2.29) and (2.30). Thus, it is trivial to demonstrate that for driving frequencies well below the lowest resonance frequency (i.e., maximum radius bubble), that Eq. (2.32) approaches the Wood's expression and is therefore dependent only on  $\beta$ —the effect of the bubble distribution is small for frequencies below bubble resonance. However, in the frequency range near the bubble resonance, the multi-radial distributions tend to spread the resonance minimum and the broad peak of the phase speed curve over a wider range of frequencies. This is clearly illustrated in Figures 2.6, 2.7, and 2.8 for which the phase speed for the bubble radius distributions above are depicted.

$\eta_m(\%) \mid R_{o,m}(\mu\mathrm{m})$	
$R_{o,m}(\mu\mathrm{m})$	
100	
200	
300	
400	
500	
600	
700	
800	

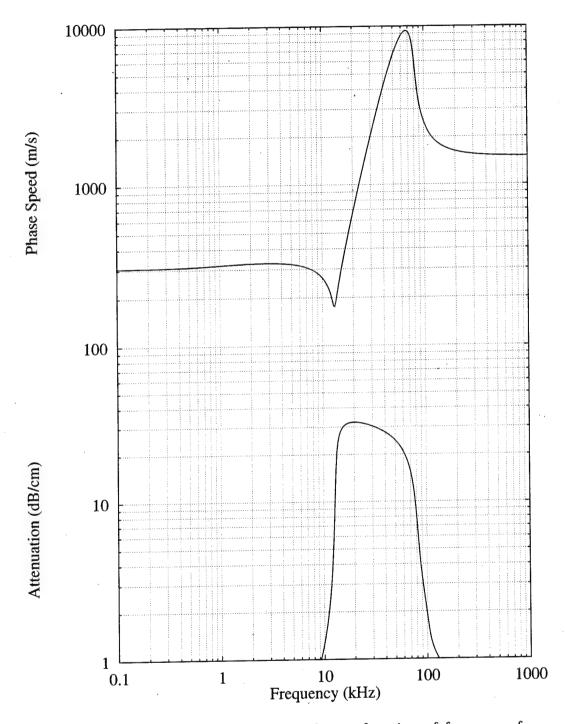
**Table 2.3:** Discrete bubble size distribution used in Figure 2.6. The total volume fraction is  $\beta = 10^{-3}$ . The percentage of bubbles (by count) at each radius are listed in columns  $\eta_m$  and  $R_{o,m}$ , respectively.



**Figure 2.6:** Phase speed as a function of frequency for a discrete bubble radius distribution. The average void fraction is  $\beta = 10^{-3}$ , and the bubble distribution is listed in Table 2.3.



**Figure 2.7:** Phase speed as a function of frequency for a bubble radius distribution having  $f(a) = \eta_o a^{-4.8}$ . The void fraction is  $\beta = 10^{-4}$ ; and the limits of the distribution is  $30 \, \mu \, \text{m} \le a \le 240 \, \mu \, \text{m}$ .



**Figure 2.8:** Attenuation and phase speed as a function of frequency for a truncated Gaussian bubble radius distribution. The void fraction is  $\beta=10^{-4}$ ; the mean bubble size is  $R_o=60\,\mu\,\mathrm{m}$ ; the standard deviation is  $\sigma_{R_o}=60\,\mu\,\mathrm{m}$ ; and the range of the distribution are  $30\,\mu\,\mathrm{m} \leq a \leq 240\,\mu\,\mathrm{m}$ .

#### 2.4 Discussion and Summary

In this chapter the collective oscillations model for sound propagation in bubbly mixture was discussed. Three approaches were presented (the first in the introduction) which led to a good approximation for the speed of sound in the two-phase mixture at frequencies below the bubble resonance frequency. For this limiting case, the speed of sound in the mixture is primarily governed by the void fraction  $\beta$ . In the first two models, the sound speed in the quasi-static limit for collective bubble oscillations was derived—recall Eq. (2.8). It is clear then, that for the extremes of the void fraction  $(\beta \to 1 \text{ or } \beta \to 0)$  that the speed of sound in the mixture approaches the values of the sound speed in the gas  $c_{\rm s}$  or the speed of sound in the liquid c, respectively. However, for  $\beta$  in between these limits, the mixture sound speed can be lowered to values that fall well below the sound speed in the liquid, and even to values below the speed of sound in the gas. This indeed is remarkable! Since the wavelength of the sound at these frequencies is much greater than the bubble radii and their respective spacing, the mixture can be treated as an effective fluid having a density near that of water and a sound speed that is dependent only on the void fraction and not on the details of the bubble population statistics.

In the first two models, the dispersive effects of the bubbles in the mixture was not considered. However, it is well known that bubbles are simple harmonic oscillators which resonate at a frequency given by the Minneart expression (Eq. 2.21). In addition to resonance phenomena, the bubbles also attenuate sound significantly. The original model for a dispersive mixture sound speed was presented by Spitzer in *Physics of Sound in the Sea* (NDRC,

1946). A more mathematical approach was taken by Commander and Prosperetti (1989), in which they derived the frequency dependence of the sound speed in the mixture which includes attenuation due to acoustic, thermal, and viscous damping. Their model builds on previous work by considering the thermomechanical properties of bubbly fluids more rigorously than before.

There are three primary frequency regimes of interest in the collective oscillations theory: below resonance, near resonance, and above resonance. Below the fundamental resonance frequency of the bubbles the mixture sound speed asymptotes to the Wood's expression (Eq. 2.9), which is dependent only on the void fraction. In this limit the mixture can be considered an effective medium with density  $\rho_e$  and sound speed  $c_e$  given by the Wood's expression (Eq. 2.9). The imaginary part (attenuation) is small in this region and is generally neglected in most studies. As the driving frequency approaches the fundamental resonance frequency of the constituent bubbles, the sound velocity in the mixture drops sharply and then rebounds to values well above the speed of sound in the host fluid. A corresponding increase in attenuation is also observed. Following the bubble resonance, the sound speed approaches its bubble free value in the liquid. The roll-off into this regime displays a marked dependence on the void fraction (i.e., for larger  $\beta$  the peak in the phase speed curve broadens). Phenomenologically, the collective oscillations model can be explained by considering the bubble as a simple resonance oscillator. The bubble oscillates in phase with the driving force below resonance, and out of phase with the driving force above resonance. In the higher frequency regime, the driving force 'sees' a much stiffer fluid medium, and hence a higher sound

speed.

In the effort described in this dissertation, the ultimate goal is to consider the scatter from spherical clouds of bubbles, primarily at frequencies below the bubble resonance. Hence, the asymptotic speed of sound given by the Wood's expression is of particular importance. This corresponds to the 'flat' region below the resonance frequencies of the largest bubbles. In this limit, the bubbly mixture can be modeled as a homogeneous fluid (Eq. 2.23) in which the effective sound speed is governed by  $\beta$  alone (Eq. 2.9) and not the individual bubbles. It is clear that this is not exactly true if one were to consider attenuation and the slight dispersive effects below bubble resonance—recall Figures 2.3, 2.4. However, the Wood's expression (Eq. 2.9) is often used since the difference in sound velocity between this approximation and that given by Eq. (2.23) is generally only a few meters per second and the attenuation is generally of negligible concern below the bubble resonance.

Several experimental efforts aimed at measuring the phase speed of sound and attenuation in a bubbly mixture have been conducted over the past fifty years. Most notable are those by Silberman (1957), and more recently by Cheyne et al. (1995). The excellent agreement with the model suggests that the bubbly bubbly fluid behaves as a homogeneous medium, especially well below the resonance frequencies of the individual bubbles. In the past decade, these ideas were independently (and vigorously) promoted by Carey (1985) and Prosperetti (1985) as a possible mechanism to explain low-frequency ambient noise emissions during wave breaking at sea. When a bubble cloud is created by a breaking wave or in a laboratory by a bubble plume generator, a

'self-induced' damped resonant oscillation results in which the eigenfrequency is determined by the phase speed within the effective medium and the characteristic length scale of the cloud (Oğuz, 1994; Yoon et al., 1991; Nicholas et al., 1994; Roy et al., 1991; Carey et al., 1993; Kolaini et al., 1994). The low-frequency noise associated with bubble cloud formation is driven in part by the broad-band 'pinch-off' noise generated by a newly formed bubbles as they attempt to reach equilibrium shape and size, and in part by hydrodynamic flow associated with the air entrainment and cloud formation process (Oğuz et al., 1995).

# Chapter 3

# Acoustic scattering from a spherical bubble cloud: Theoretical model

#### 3.1 Introduction

Scattering can be regarded as radiation from a 'source' which is driven by an incident wave rather than by a local external generator. As an example, consider a plane wave traveling in the z-direction through a uniform fluid. A volume element of the fluid will be set into oscillatory motion in the direction of wave propagation, and it will be subject to periodic compression and rarefaction. The response of the fluid to the incident wave will involve both the inertial mass and the compressibility of the fluid. The volume element in the fluid responds by pulsating and oscillating, giving rise to both monopole and

dipole radiation.

For a uniform fluid, all volume elements represent sources of equal strength but different phases, where the phase is determined by the position of the element along the propagation path. A wave front is defined by all volume elements oscillating in equal phase. Ahead of the wave front sound is propagated by each volume element which radiates sound in phase (the so called Huygens' 'wavelets'), and behind it the radiation is 180° out of phase resulting in zero backwards propagation.

Now consider an inhomogeneous volume embedded within the previously uniform fluid. The foreign body possesses a different inertial mass and/or compressibility than that of the host fluid. The plane wave, as before, travels in the z-direction giving rise to Huygens' wavelets, but when the wavefront reaches the boundary of the inhomogeneous region the differing material properties create virtual sources having different strength and phase compared to their neighbors in the wavefront. This results in an excess or scattered wave contribution radiating from the foreign body. For non-uniformities within the wave front that are small in comparison to the wavelength (ka < 1), the monopole field will be proportional to the difference in compressibility and the dipole field proportional to the density contrast.

In this section a theoretical model for acoustical scattering from a penetrable sphere (target) in the free-field will be considered. Free-field implies that the host fluid is free of inhomogeneities or reflecting surfaces, with the exception of a single embedded target. Although the amplitude, periodicity, and phase of the incident plane waves are known, the exact driving force

causing the sphere to radiate sound is not, and therefore must be calculated exactly. For wavelengths much larger than the target radius  $(\lambda \gg 2\pi a)$ , the scatterer is considered 'compact' and the driving pressure on the boundary is often assumed to be the incident field (i.e., the Born approximation). This results primarily in monopole and dipole scattered contributions. However, for shorter wavelengths, the driving pressure on the surface of the sphere will not be uniform and the boundary value problem must be solved directly. In special cases, the sound waves penetrating the scatterer or traversing its boundary coincide with the eigenvibrations of the sphere and interfere constructively by setting up self–sustaining standing waves in the object or on its surface. It is intuitively clear that when the incident wave oscillates at one of the target's eigenfrequencies that the corresponding eigenvibration in the sphere will be excited and strongly affect the scattered wave.

Our interest is in scattering from bubble clouds in the ocean. Although it has long been known that a two-phase mixture of air bubbles in water pulsates in a collective fashion and that the sound speed in the mixture below the bubble resonance frequency is determined by the void fraction alone and therefore the region containing the mixture can be treated as an effective medium, it was not proposed until 1985 that such a mixture could serve as a simple radiation source and a scatterer of sound. When a bubble cloud is created by a breaking wave, a damped resonant oscillation results in which the eigenfrequency is determined by the phase speed within the effective medium and the characteristic length scale of the cloud (Carey & Browning, 1988; Oğuz, 1994). Several examples of 'self-induced' sub-kilohertz radiation from

bubbly mixtures were given in Chapter 1 and Chapter 2 (Yoon et al., 1991; Carey et al., 1993; Nicholas et al., 1994; Roy et al., 1991; Nicholas et al., to be submitted). It seems plausible then that if a bubbly mixture is an efficient radiator of sub-kilohertz sound, that it might also serve as effective scattering target. This idea was explored in a recent experiment performed in a fresh water at Lake Seneca in New York state (Roy et al., 1992) and is the impetus for the scattering model presented in this chapter.

#### 3.2 Formalism

In this model, a continuous stream of monochromatic plane waves incident on a compressible sphere of radius a surrounded by an infinite fluid as indicated in Figure 3.1 is assumed. The density and sound speed within the sphere are denoted by  $\rho_e$  and  $c_e$ , respectively. In the surrounding fluid, these quantities are given by  $\rho$  and c. This solution is not limited to bubble clouds, but for the purposes of this work the effective medium approximations derived in Section 2 will ultimately be applied.

The solution for scattering from a sphere is straightforward. Anderson (1950) and Rshevkin (1963) treat the case for scattering from a fluid sphere. Morse and Ingard (1968) arrive at the solution after an extremely mathematical analysis using an integral Green's function technique. Morse and Feschbach (1953) used an elegant phase shift analysis which is better suited to spheres having an acoustic coating of known impedance or more complex geometries. A comprehensive survey of acoustic and electromagnetic scattering from sim-

ple shapes (i.e., separable geometries) such as spheres, cylinders, spheroids, cones, plates, etc., can be found in (Bowman et al., 1987)—much of this work is based on the previous solutions by (Morse & Feshbach, 1953). Flax et al. (1981) discuss resonance scattering theory and its application to solid spherical inclusions, air bubbles, and various fluid loaded shells. Their work departs from the previous in that they apply broad band (short pulse lengths) techniques, rather than the usual monochromatic (steady state or CW) assumptions, using Fourier analysis. All the works cited above, however, arrive at identical scattering solutions for the fluid sphere case<sup>1</sup>. However, Anderson's approach is preferred due to its inherent simplicity (one should note that his scattering angle is shifted 180° from the presently accepted notation).

An alternative approach taken by Gragg and Wurmser (1993) was to numerically solve for the backscattered wave from a bubble filled sphere or other shape. They implemented a Boundary Value Integral (BVI) technique to computationally solve for the scattered wave contribution from a bubble cloud (both spherical and spheroidal) by applying the appropriate boundary conditions on a boundary 'mesh' enclosing an arbitrarily shaped scatterer. This technique is shown to result in the exact scattering solution for the sphere and spheroid given an appropriate tiling, but is better suited to scattering from more complex geometries since the numerical method is computationally expensive. The advantage to the method presented in this chapter, on the other hand, is that it is simple and straightforward, and therefore can be used to efficiently predict the scattering spectrum from a spherical bubble cloud.

<sup>&</sup>lt;sup>1</sup>This can be shown in particular for the solutions by Anderson, Rschevkin, and Morse and Ingard after much algebraic work.

It will be shown that the solution for acoustic scattering from a fluid sphere has a marked dependence on the internal sound speed and density. Thus, in order to consider the case of scattering from bubble clouds the collective oscillations model will be applied as presented in Chapter 2. Scattering from a spherical bubble cloud will be considered explicitly following a review of sound scattering from a 'hard' impenetrable sphere and an idealized (lossless) bubble. These cases are well known and they are discussed only for comparison and completeness. Furthermore, a simple analytic expression for the scattering response in the low frequency limit (i.e., small ka) will be derived as it proves to be a fundamentally useful tool and yields considerable physical insight. In addition, this expression provides a good approximation for the monopole, or 'breathing mode,' resonance frequency of the sphere.

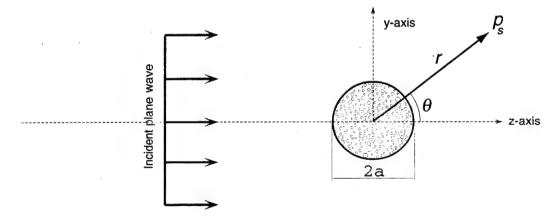
In this modeling effort, the sphere is chosen as the scattering target not because spherical bubble clouds are created beneath the ocean surface, but rather because the length scales of the clouds created are small in comparison to the wavelength of sound over the frequency range of interest (i.e., the sub-kilohertz frequencies where long range sonar systems operate). The natural creation of bubble clouds results in a densely populated region of bubbly fluid having a characteristic size of  $0.5\,\mathrm{m}$  or less in length scale. For low acoustic frequencies, such a target is termed acoustically compact, that is ka < 1. For most acoustic wavelengths encountered in this study, the cloud can be considered 'compact' since  $\lambda \geq 2\pi a$ , and the response of the cloud will be dictated by the monopole mode which is primarily dependent on the volume and compressibility of the target, and more or less independent of shape pro-

vided that the aspect ratio is close to unity. For example, consider an acoustic frequency of 300 Hz. The sonic wavelength in seawater is  $\lambda \approx 5\,\mathrm{m}$ , greater than the characteristic dimensions of the target ( $a=0.5\,\mathrm{m}$ ). The scatterer is well characterized by a sphere and the solution is cast in a multipole expansion (Lighthill, 1975). The most important term in describing the low–frequency behavior of the system is the monopole term.

The coordinate system for the sphere is shown in Figure 3.1, where the relationship between the Cartesian and spherical polar coordinate systems is given by the usual transformations

$$x = r \sin \theta \cos \phi, \quad y = r \sin \theta \sin \phi, \quad z = r \cos \theta,$$
 (3.1)

and the coordinate origin is located at the center of the sphere.



**Figure 3.1:** Scattering geometry. The z-axis is aligned with the direction of plane wave propagation. The coordinate system origin is located at the center of the sphere.

The sphere is assumed to consist of an isotropic fluid material of density  $\rho_e$  and sound speed  $c_e$ . Outside the sphere, an infinite fluid of density  $\rho$  and

sound speed c exists. A continuous plane wave of angular frequency  $\omega$  insonifies the sphere from the z-direction. The time and spatial dependence of the incident sound is  $\exp[i(\mathbf{k}\cdot\mathbf{r}-\omega t)]$ , where the magnitude of the wave number is  $k = \omega/c = 2\pi/\lambda$ , and  $\lambda$  is the wavelength of the incident sound in the host fluid. Inside the sphere, the wave number is simply  $k_e = \omega/c_e = 2\pi/\lambda_e$ , and  $\lambda_{\epsilon}$  is the effective wavelength of sound inside the sphere.

The general solution for the pressure everywhere is

$$p = \begin{cases} p_i + p_s & r \ge a, \\ p_{\text{int}} & r < a, \end{cases}$$
(3.2)

where the subscripts i, s, and int, refer to the respective incident, scattered, and internal fields. A monochromatic incident plane wave of amplitude  $p_o$ impinging on the sphere has the form

$$p_{i} = p_{o}e^{i(\mathbf{k}\cdot\mathbf{r}-\omega t)}$$

$$= p_{o}e^{i(kr\cos\theta-\omega t)}.$$

$$(3.3)$$

$$= p_o e^{i(kr\cos\theta - \omega t)}. \tag{3.4}$$

This disturbance gives rise to both internal and external waves,  $p_{\text{int}}$  and  $p_s$ , respectively.

Exploiting the symmetry of the problem, the incident field can be expressed as a series expansion in spherical harmonics dependent on r and  $\theta$ , angular frequency  $\omega$ , wave number k, and time t.

$$p_i(\mathbf{r},t) = p_o \sum_{m=0}^{\infty} (2m+1)i^m P_m(\cos\theta) j_m(kr) e^{-i\omega t}, \qquad (3.5)$$

where the  $j_m(kr)$  is the spherical Bessel function of the first kind of order m; and  $P_m(\cos \theta)$  is the Legendre polynomial of order m. The solution is independent of the angular direction  $\phi$  due to azimuthal symmetry.

At the boundary of the sphere (r = a), both the pressure and normal component of the particle velocity must be continuous. For the acoustic pressure, the boundary condition is

$$p_i(a) + p_s(a) = p_{\text{int}}(a).$$
 (3.6)

The continuity of radial component of the particle velocity on the spherical boundary is

$$u_{i,r}(a) + u_{s,r}(a) = u_{\text{int},r}(a).$$
 (3.7)

The relationship between the pressure and the normal velocity is

$$u_{\star,r} = \frac{-i}{\rho_{\star}c_{\star}} \left[ \frac{\partial p_{\star}}{\partial (kr)} \right], \tag{3.8}$$

where the subscript  $\star$  denotes the incident, internal, or scattered solutions. In addition, the pressure inside the sphere must be finite everywhere, and outside the sphere the amplitude of the scattered pressure disturbance must go to zero

as r approaches infinity.

The acoustic pressure must satisfy the three-dimensional wave equation:

$$\nabla^2 p - \frac{1}{c^2} \frac{d^2}{dt^2} p = 0, (3.9)$$

which simplifies to,  $\nabla^2 p + k^2 p = 0$ , after evaluating the time derivative.

The solution for the internal and scattered pressure are given by the partial wave expansion in spherical harmonics as follows:

$$p'(r < a, t) = \sum_{m=0}^{\infty} A_m P_m(\cos \theta) j_m(kr) e^{-i\omega t}. \tag{3.10}$$

$$p_s(r > a, t) = \sum_{m=0}^{\infty} B_m P_m(\cos \theta) \begin{bmatrix} j_m(kr) \\ n_m(kr) \end{bmatrix} e^{-i\omega t}.$$
 (3.11)

Here, the  $j_m(x)$  and  $n_m(x)$  represent the spherical Bessel functions of the first and second kind of order m and argument x (usually called the spherical Bessel and spherical Neumann functions, respectively). The coefficients  $A_m$  and  $B_m$ are obtained by applying the boundary conditions Eq. (3.6) and (3.7). Our interest is in the scattered field, not the internal pressure field, thus

$$p_s(r) = -\frac{p_o}{2} \sum_{m=0}^{\infty} i^m (2m+1) \mathcal{R}_m P_m (\cos \theta) h_m(kr) e^{-i\omega t},$$
 (3.12)

where  $h_m(x) = j_m(x) + in_m(x)$  is the spherical Hankel function, and  $\mathcal{R}_m$  satisfies the boundary conditions. Individual terms in the expansion Eq. (3.12) yield the monopole, dipole, quadrupole, etc., amplitudes (i.e.,  $m = 0, 1, 2, \ldots$ ,

respectively). The monopole term radiates sound uniformly in all directions since  $P_0(\cos \theta) = 1$ , identically. Furthermore, the m = 0 term represents the volume or breathing mode of the sphere. Unless otherwise specified, the time dependence of the solution  $\exp(-i\omega t)$  will be discarded.

The coefficient  $\mathcal{R}_m$  in Eq. (3.12) is determined from the boundary conditions where,

$$\mathcal{R}_{m} = \frac{\rho_{e} c_{e} j_{m}(k_{e} a) j'_{m}(k a) - \rho c j'_{m}(k_{e} a) j_{m}(k a)}{\rho_{e} c_{e} j_{m}(k_{e} a) h'_{m}(k a) - \rho c j'_{m}(k_{e} a) h_{m}(k a)},$$
(3.13)

and  $j'_m(x)$  and  $h'_m(x)$  are the derivatives of the spherical Bessel and Hankel functions.

In the far field  $(kr \gg 1)$ , the scattered pressure in Eq. (3.12) asymptotes to

$$p_s(\mathbf{r}) \longrightarrow i \frac{p_o}{2} \frac{e^{ikr}}{kr} \sum_{m=0}^{\infty} (2m+1) \mathcal{R}_m P_m (\cos \theta),$$
 (3.14)

$$\longrightarrow p_o \frac{a}{2r} \mathcal{F}(ka, \theta),$$
 (3.15)

where  $\mathcal{F}(ka, \theta)$  is the so called dimensionless scattering form function.

The preceding equation (3.14) deserves special attention. The amplitude of the solution falls of inversely with distance as expected from a spherical radiating source. Also, each term in the series has a strong dependence on the internal and external densities and sound speeds, as well as the driving frequency (this is because  $k = 2\pi f/c$  and  $k_e = kc/c_e$ ). In addition, the angular dependence of the scattered radiation is given by the Legendre polynomials

which result in preferential radiation in some directions since the boundary of the target is not driven uniformly. While it can be shown that the directionality is only of moderate importance for small ka and can be quite significant for larger ka, our interest is in the backscattered contribution. Accordingly, a further simplification by setting  $\theta = 180^{\circ}$  will be applied and the Legendre polynomials simplify to the well known relation  $P_m(-1) = (-1)^m$ .

The coefficient  $\mathcal{R}_m$  contains the fundamental dependence of the scattering on frequency and material properties of the target and its host. It is a complex quantity, and thus the magnitude and phase are important. For a given density and sound speed of the host and target, resonances or peaks in the scattering spectrum will occur for singularities of the coefficient  $\mathcal{R}_m$  (i.e., when the denominator of Eq. (3.13) approaches zero). This will be discussed further in Section 3.4

## 3.3 Backscattering form function

Although the incident and scattered pressures (Eq. (3.4) and (3.14)) are experimentally measurable quantities, it is sometimes impractical and often unnecessary to report absolute levels when discussing the efficiency of the scattering by a target. Rather, coefficients of the relative scattering strength, such as the scattering form function, scattering cross section, ratio of pressure amplitudes, and target strength are generally reported.

The total scattering cross section  $(\sigma_s)$  is defined by the ratio of the total power scattered divided by the incident intensity. Expressed as an integral over

all angles, the total scattering cross section is

$$\sigma \equiv \frac{\Pi_s}{I_i} = \iint_S \frac{d\sigma_s}{d\Omega} d\Omega, \qquad (3.16)$$

where the quantity  $\Pi_s$  is the total power scattered over all angles;  $d\Omega$  is the increment of solid angle; and  $\frac{d\sigma_s}{d\Omega}$  is the differential scattering cross section. When  $\sigma$  is evaluated, it determines the power scattered per unit solid angle and per unit incident intensity. The differential scattering cross section has the dimensions of area is defined as

$$\frac{d\sigma_s}{d\Omega} \equiv \frac{I_s r^2}{I_i},\tag{3.17}$$

where  $I_s$  is the scattered intensity,  $I_i$  is the incident intensity, and r is the distance from the point of interest (the receiver) to the target (Clay & Medwin, 1977). For an isotropic scatterer,  $d\sigma_s/d\Omega$  is independent of direction and equal to  $\sigma_{\rm bs}/4\pi$ , where  $\sigma_{\rm bs}$  is the backscattering cross section (i.e.,  $\sigma_s$  evaluated at  $\theta = 180^{\circ}$ ).

Closely related to the backscattering cross section is a quantity called the relative target strength level (TS) defined by (Urick, 1967)

$$TS = 10 \log_{10} \left(\frac{I_s}{I_i}\right)_{r=1 \text{ m}},$$
 (3.18)

where the incident and scattered intensities are evaluated at a 1 m distance.

In the sonar community, the TS is the standard quantity used to indicate the scattering strength of an object. If you consider incident plane waves and isotropic scattering, then Eq. (3.17) and (3.18) combine to yield

$$TS = 10 \log \left(\frac{\sigma_{bs}}{4\pi r^2}\right)_{r=R_{ref}}.$$
(3.19)

Further discussion of the TS and its application to scattering measurements is outlined in Appendix A.

Since acoustic intensity is proportional to the square of the pressure amplitude for both planar and spherical waves, the scattering cross section and the target strength are closely related to the pressure amplitudes in Eq. (3.14). Thus, the relationship between the backscattering cross section and the scattered and incident pressure amplitudes is given by

$$\frac{\sigma_{\rm bs}}{4\pi} = \frac{|p_s|^2}{|p_i|^2} r^2,\tag{3.20}$$

and the magnitude of the dimensionless scattering form function is

$$|\mathcal{F}| = \frac{2r}{a} \frac{|p_s|}{|p_i|},\tag{3.21}$$

and is dependent explicitly on ka, g, and h. The form function is a dimensionless quantity which describes the strength of a scatterer as a function of ka. Now,

$$TS = 20 \log_{10} \left( \frac{a}{2r} |\mathcal{F}| \right)_{r=1 \text{ m}} = 20 \log_{10} \left( \frac{|p_s|}{|p_i|} \right)_{r=1 \text{ m}},$$
 (3.22)

in contrast to Eq. (3.21), is not explicitly dependent on ka. The radius (a) and

the frequency must be known prior to computation of the TS.

To simplify the analysis, let us define the relative quantities  $g \equiv \rho_e/\rho$ , and  $h \equiv c_e/c$ , and rewrite the scattered pressure series solution (Eq. 3.14) in terms of these quantities. Then the magnitude of the backscattering form function is

$$|\mathcal{F}| = \left| \frac{1}{ka} \sum_{m=0}^{\infty} (-1)^m (2m+1) \mathcal{R}_m \right|,$$
 (3.23)

and

$$\mathcal{R}_m = \frac{j_m(\frac{ka}{h})j'_m(ka)gh - j'_m(\frac{ka}{h})j_m(ka)}{j_m(\frac{ka}{h})h'_m(ka)gh - j'_m(\frac{ka}{h})h_m(ka)}.$$
(3.24)

It is important to stress that so long as the material properties of the scatterer and host are not dispersive, then the form function is strictly parameterized by the Helmholtz number, ka, and the material quantities g and h. This scaling between size and frequency is common to many problems in acoustics, and means that neither the scatterer size nor the frequency need to be known independently in order to solve the problem.

From this point forward, all references to the scattering strength will be made either through the target strength or scattering form function. Generally, the form function will be reserved for use when comparing the theoretical backscatter from targets having a non-dispersive interior (and exterior) sound speed. For such cases, calculation of the backscattering form function as a function of ka alone both generalizes and simplifies the analysis and calculations considerably.

On the other hand, for a bubbly mixture the velocity of sound is dependent of the acoustic frequency as shown in Chapter 2. Furthermore, the dispersion relation for the mixture sound speed is in no way dependent on the cloud dimensions, much less ka. Therefore, a plot of the backscatter form function versus ka has little meaning for a dispersive medium unless either the cloud radius or frequencies are specified. Of course, this is not the case for non-dispersive media, where for a given sound speed and density ratio, h and g, respectively, a single calculation of the form function yields valid results for any range of frequencies or sphere radii (i.e., ka) considered.

To calculate the TS, the cloud radius and frequency range must be specified a priori. The TS will be used to compare calculated predictions with experimental measurements. In an experiment the characteristic dimensions and acoustic parameterizations (g and h) of the target are not needed to measure the TS.

Initially, the calculations were performed using Mathematica. Once it was determined that the calculations were accurate (by comparison with the plots in Anderson's (1950) paper), the codes were rewritten in the C/C++ programming languages and integrated with the collective oscillations code. The calculations of the special functions (spherical Bessel, spherical Neumann, and Legendre polynomials) were performed using the standard recursion relations (simple transcendental equations involving sin and cos). Since the interest in this research is confined to the lower order scattering modes (i.e., small ka), no special attention was paid to the accuracy of the calculation of these special functions.

#### 3.3.1 Case Studies

The dependence of the backscatter form function of a sphere in water as a function of ka is now explored for three cases of different material parameters. Consider first the case of the idealized hard impenetrable sphere. In this case, the internal acoustic impedance is given by  $Z_e = \rho_e c_e \to \infty$ . Thus in Eq. (3.24) the quantity  $gh \to \infty$  and Eq. (3.23) reduces to

$$|\mathcal{F}| = \left| \frac{1}{ka} \sum_{m=0}^{\infty} (-1)^m (2m+1) \frac{j'_m(ka)}{h'_m(ka)} \right|. \tag{3.25}$$

The form function versus ka for the hard sphere is shown in Figure 3.2, where the resonances or peaks of the system are governed by the minima of  $h'_m(ka)$ . The peaks are due to interference between waves propagating along the fluid boundary layer on the circumference of the sphere (a phenomenon known as creeping waves) and the specular reflection. For increasing ka, the amplitude of the oscillations in the form functions diminish resulting in perfect specular reflection. And as such, the magnitude of the form function approaches unity. For specular reflection from a hard sphere the TS in the high frequency (short wavelength) limit asymptotically approaches  $TS = 20 \log_{10}(a/2)$ , and thus a 2 m radius sphere will possess a TS=0 dB. Consequently, hard metallic spheres are often used as calibrated targets.

For the case of an idealized bubble submerged in water, that is an air filled sphere without damping, the acoustic parameterizations are g = 0.001 and h = 0.22. In Figure 3.3, the backscattering form function as a function of ka for a bubble is illustrated. Here, there is an obvious very sharp resonant

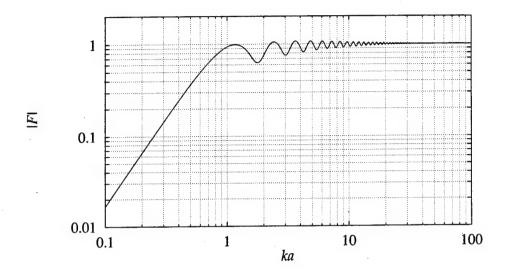


Figure 3.2: Backscattering form function versus ka for an impenetrable sphere.

peak in the spectrum at ka = .0134, followed by a rather smooth response and then several 'spiky' resonances at higher ka. The fundamental resonance corresponds to  $f_o = 0.0134 \, c/2\pi a \approx 3.2/a$  in Hz, where the radius a is given in meters. Remarkably, this peak corresponds exactly to the well know Minneart frequency given in Eq. (2.21).

The third case is that of a frequency independent low-sound velocity bubble cloud as shown in Figure 3.4. In this case the void fraction within the cloud is  $\beta = 10^{-3}$  and the Wood approximations yields  $c_{\epsilon} \approx 313 \, \text{m/s}$ . Accordingly, g = 0.999 and h = 0.208. Again, a peak is observed in the spectrum for ka < 1. Although the sound velocity ratio in this case and the previous case for the bubble are nearly identical, the resonance response is quite different. This suggests that the density ratio plays an important role in both the frequency and width of the resonance peak. We would like to have the ability to predict the resonance frequency for all target spheres. As a first

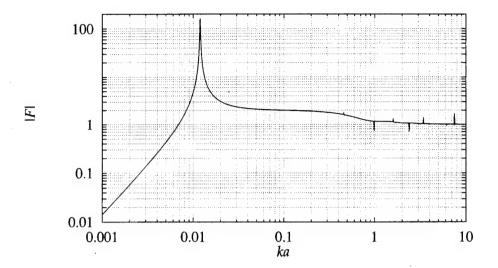


Figure 3.3: Backscattering form function versus ka for a bubble.

'guess', we begin with the Minneart relation,  $f_o = (1/2\pi a)\sqrt{3\gamma P_{\infty}/\rho}$  for the natural resonance of a bubble, and recall that the velocity of sound in the gas is approximately  $c_{\rm g} = \sqrt{\gamma P_{\infty}/\rho_{\rm g}}$ . Substitution of the gas sound speed into the former equation results in

$$f_o = (c_e/2\pi a)\sqrt{3\rho_e/\rho},$$
 (3.26)

where the subscripts for the gas are replaced by the subscript denoting the generalize sphere. From Eq. (3.26), we have

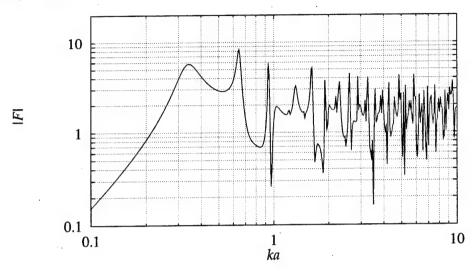
$$ka = 2\pi a f_o/c \tag{3.27}$$

$$= (c_e/c)\sqrt{3\rho_e/\rho} \tag{3.28}$$

$$= h\sqrt{3g} \tag{3.29}$$

at resonance. For the case of the bubble cloud considered above, this suggests

that ka = 0.36. This appears to be a good approximation since the peak actually occurs at ka = 0.355.



**Figure 3.4:** Backscattering form function versus ka for a bubble cloud having  $\beta=10^{-3}$ , thus  $c_e\approx 313\,\mathrm{m/s}$  (i.e., g=0.999 and h=0.208).

The fundamental observation is that for a compressible fluid sphere, the monopole resonance is clearly indicated by the first peak in each of the curves. For  $ka \ll 1$ , all the cases examined exhibit a similar asymptotically increasing slope—classically termed Rayleigh scattering. Beyond the monopole resonance (ka > 1), in the regime known as geometrical scattering, the complex series of higher order peaks are due to internal resonances and can be explained by resonance scattering theory. These resonances (eigenmodes) are governed by minimizing the denominator of  $\mathcal{R}_m$ .

In the preceding development, the internal sound velocity of the sphere was not dispersive and thus scattering form function is dependent on ka rather than frequency explicitly. In order to consider the dispersive effects of a true bubbly fluid (using the collective oscillations model), a particular sphere radius

must be chosen to consider the frequency dependence. Recall, from Chapter 2 the discussion of the mixture sound speed in a bubble cloud having void fraction  $\beta=10^{-3}$  with a mean bubble size of 500  $\mu$  m. From Figure 2.3, it is clear that the mixture sound velocity asymptotes to the Wood approximation is valid for all frequencies below 2 kHz and departs from this limiting value when evaluated at higher frequencies. The attenuation over this frequency range is not negligible. In the region surrounding the bubble resonance, the attenuation in the bubbly mixture is great and thus the sphere cannot be expected to scatter sound in the same manner as it would at the lower frequencies.

In Figure 3.5, the TS as a function of frequency, is plotted for a bubble cloud having radius  $a=0.25\,\mathrm{m}$ , void fraction  $\beta=10^{-3}$ , and a uniform bubble distribution of radii  $R_o=500\,\mu\,\mathrm{m}$ . Also plotted is the TS of an individual  $500\,\mu\,\mathrm{m}$ -radius bubble. The similar pair of curves represents the effect of using the Wood equation (lighter curve) as compared to the dispersive sound speed formalism (darker curve) obtained using the collective oscillations theory in Eq. (2.25). It is clear that introducing attenuation has reduced the overall scattering level at the lowest frequencies, as well as shifted the higher order peaks to lower frequencies. The solid curve with a single peak near  $5.5\,\mathrm{kHz}$  represents the backscatter form function of a bubble having radius  $R_o=500\,\mu\,\mathrm{m}$ . Just above the monopole resonance of the constituent bubbles, the TS of the dispersive bubble cloud approaches that of the perfectly reflecting impenetrable sphere since the interior sound speed is increased dramatically (recall Figures 2.3 and 3.2).

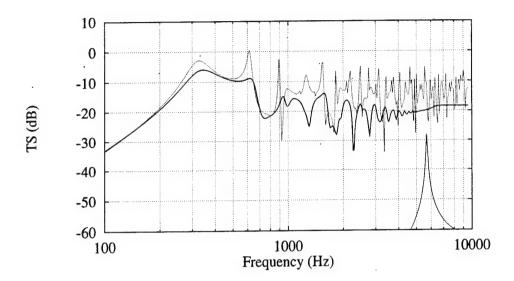
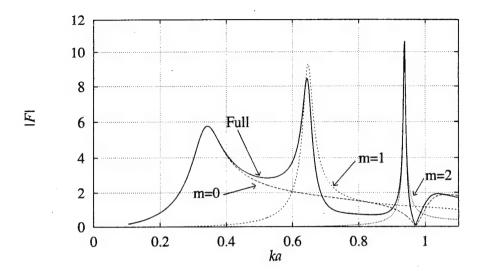


Figure 3.5: TS versus frequency. The upper two curves are for a cloud radius of  $a=0.25\,\mathrm{m}$  and  $\beta=10^{-3}$ . In the darkest solid curve, the dispersive internal sound velocity was used, and in the dashed curve the internal sound speed was calculated using the Wood expression (recall Figure 3.4). The thin solid line corresponds to the TS of single  $500\,\mu\,\mathrm{m}$  radius bubble. Recall from Eq. (3.22) that a perfectly reflecting sphere ( $|\mathcal{F}|=1$ ) having radius  $a=2\,\mathrm{m}$  possesses TS = 0 dB.

### 3.4 The low-frequency limit

From Figures 3.3 and 3.4, it is clear that the first or 'breathing-mode' resonance occurs in all cases for ka < 1. It would be useful to explore the behavior of the form function as a function of ka for a few of the lower order modes. In Figure 3.6, the m = 0, 1, 2 order terms of the backscattering form function (Eq. 3.23) are plotted along with the full solution. It is clear that the m-th order peak in the partial-wave solution arise as a direct consequence of m-th order term in the expansion of Eq. (3.23).

Also, we should remark that the lowest order peak is dependent solely on the m=0, monopole term, of the series expansion. For the lowest order peak,



**Figure 3.6:** Modes of sphere with g = 0.999 and h = 0.208 plotted against ka.

an approximation to the form function is obtained by expanding the m=0 term of the series (Eq. 3.24) to lowest surviving order in ka. Small argument approximations to the spherical Bessel and Neumann functions (since  $h_m(x) = j_m(x) + in_m(x)$ ) are given by the following expansions:

$$j_o(x) = \left\{ 1 - \frac{\frac{1}{2}x^2}{3} + \frac{\left(\frac{1}{2}x^2\right)^2}{2!(3)(5)} - \dots \right\},$$

$$n_o(x) = \frac{1}{x} \left\{ 1 - \frac{1}{2}x^2 + \frac{\left(\frac{1}{2}x^2\right)^2}{2!(1)(3)} - \dots \right\}.$$

Upon making these substitutions and retaining only the lowest order surviving terms of ka, we find that the form function is approximated by

$$|\mathcal{F}| \approx \frac{(ka)^2}{3} \left| \frac{1 - gh^2}{\left(gh^2 - \frac{(ka)^2}{3}\right) - i\frac{(ka)^3}{3}} \right|.$$
 (3.30)

Thus, for ka < 1, the magnitude of the slope of the form function is  $(ka)^2$ , exhibiting what is known as Rayleigh scattering behavior<sup>2</sup>. The peak in this limiting form is clearly dependent on the density and sound–speed squared ratios and the monopole resonance frequency is well approximated by equating the real portion of the denominator (Eq. 3.30) to zero. Employing this approximation, it is found that

$$0 \equiv \left(gh^2 - \frac{(k_o a)^2}{3}\right) \tag{3.31a}$$

$$k_o^2 = \frac{3\rho_e c_e^2}{a^2 \rho c^2} \tag{3.31b}$$

$$\Omega_{\rm o} = \frac{c_e}{2\pi a} \sqrt{\frac{3\rho_e}{\rho}}.$$
 (3.31c)

In Eq. (3.31),  $k_o = 2\pi\Omega_0/c$  is the resonance wave number,  $\Omega_0$  is the monopole resonance frequency (in Hz) for a spherical scatterer of radius a. For a bubble, we set  $c_e = c_g \equiv \sqrt{\kappa P_\infty/\rho_g}$ ,  $\rho_e = \rho_g$ , and  $a = R_o$  so that Eq. (3.31c) simplifies to the Minneart expression. The monopole resonance frequency (Eq. 3.31c), incidentally, agrees with our previous 'guess' in Eq. (3.26).

For a bubble cloud, the sound speed is approximated by the Wood expression (Eq. 2.9) for small ka, and we find that the monopole resonance frequency is governed by a *modified* Minneart expression, namely

$$\Omega_0 = \frac{1}{2\pi a} \sqrt{\frac{3\kappa P_\infty}{\beta (1-\beta)\rho}}.$$
(3.32)

In this case, the additional approximation that  $\rho_e \approx \rho$  was also applied.

<sup>&</sup>lt;sup>2</sup>The Rayleigh regime is actually noted for a  $(ka)^4$  slope dependence, which is indeed the case when one expands the scattering cross section and not the form function.

At the monopole resonance, the ratio of scattered to incident pressure amplitudes (the argument of the TS in Eq. (3.22)) is:

$$\frac{|p_s|}{|p_i|} = \frac{a}{2r} |\mathcal{F}| \propto \frac{1}{\Omega_0 r} \left| \frac{\rho_e c_e^2}{\rho c^2} - 1 \right|. \tag{3.33}$$

As illustrated in Figure 3.7, the low frequency approximation of the form function predicts, qualitatively and semi-quantitatively, both the amplitude and resonance frequency of the scatterer. Higher order modes of the system can be solved in a similar manner (Flax et al., 1981). However, it is doubtful that equations with more physical insight than Eq. (3.32) and (3.33) will be obtained.

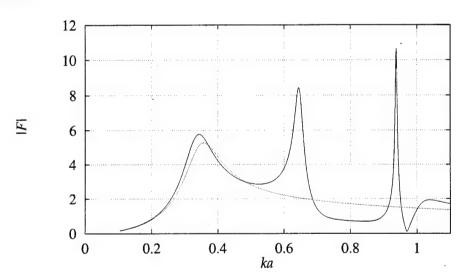


Figure 3.7: A comparison of the asymptotic form function with the exact calculation

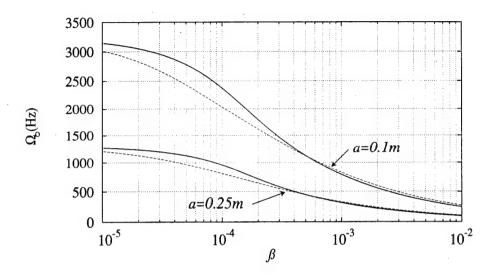
The simple analytic expressions Eq. (3.30) and (3.31c) along with the Wood equation lead to the following conclusions for scattering at resonance:

(1) For a given cloud radius, increasing the void fraction results in decreased

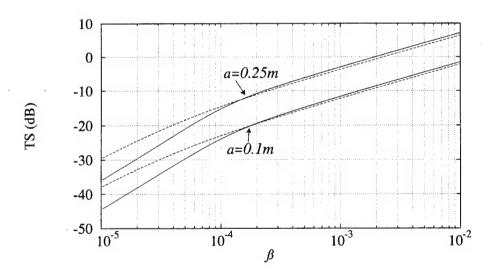
sound speed and a decrease in the monopole resonance frequency (Figure 3.8(a)). A corresponding increase in the monopole scattering amplitude is also observed (Figure 3.8(b)). (2) For a fixed volume fraction (*i.e.*, fixed sound speed), the resonance frequency decreases with increasing cloud radius (Figure 3.9(a)) and the scattering amplitude increases (Figure 3.9(b)). A particularly useful feature of these equations is their ability to predict the scattering behavior near the monopole resonance base only on the cloud radius and the void fraction. In Figures 3.8(a)-3.9(b) the solid line is calculated using the full wave expansion (Eq. 3.23) and the dashed line from the asymptotic expression (Eq. 3.30).

The asymptotic equations in Eq. (3.32) and (3.30) adequately predict the behavior of the scatterer at frequencies near the monopole peak even though the expansion used to obtain Eq. (3.33) is not mathematically valid for the  $k_e a > 1$ , as will usually be the case for a low sound speed scatterer. Perhaps the most interesting result is that using simple asymptotic expressions we can easily predict  $\Omega_0$  (Eq. 3.32), the TS surrounding the resonance frequency (Eq. 3.30), and  $c_e$  (Eq. 2.10b) to within an acceptable margin having knowledge of only the cloud radius and the void fraction. Furthermore, the monopole resonance for other cloud geometries (cylinders, spheroids, etc) should be well approximated by these simple expressions provided that the aspect ratio does not vary considerably from unity. For such cases, we need only the total gas volume in the cloud and total liquid volume enclosing the mixture to determine the equivalent radius, void fraction, and therefore determine  $\Omega_0$ , TS, and  $c_e$ .

From Figure 3.9 it is evident that a 0.1 m-radius cloud with void fraction

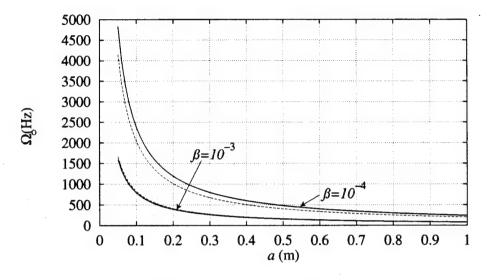


(a) Peak monopole resonance frequency.

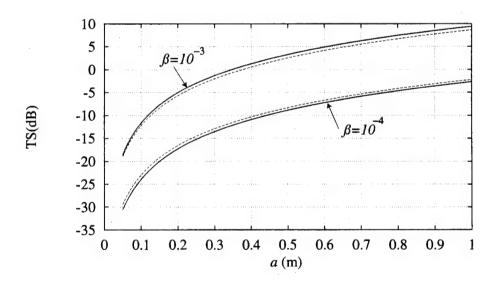


(b) Peak TS evaluated at monople resonance frequency.

Figure 3.8: Peak monopole TS and frequency as a function of void fraction for cloud radii  $a = 0.1 \,\mathrm{m}$  and  $a = 0.25 \,\mathrm{m}$  using the exact series expansion (solid) and the asymptotic expression (dashed).



(a) Peak monopole resonance frequency.



(b) Peak TS evaluated at monople resonance frequency.

**Figure 3.9:** Peak monopole TS and frequency as a function of cloud radius for a void fraction of  $\beta = 10^{-4}$  and  $\beta = 10^{-3}$  using the exact series expansion (solid) and the asymptotic expression (dashed).

10<sup>-3</sup> resonates at about 800 Hz with a target strength of about -11 dB. The latter two numbers are consistent with the observations of Adair and Huster (Figure 1.4). Furthermore, the cloud size and void fraction suggested here are not outside the realm of possibility (Monahan & Lu, 1990; Lamarre & Melville, 1992). It would appear then that this model may have some validity. To examine this hypothesis further, a carefully controlled experiment was recently conducted in a fresh water lake in New York state (see Part II of this document).

# 3.5 Discussion and Summary

In this chapter the theory for acoustic scattering from a compliant sphere in the free-field was presented. The solution for the scattered pressure at every point in the sphere's exterior is given by the normal mode expansion (Eq. 3.12). The classical solution was obtained by simply meeting the requirements on the boundary of the sphere of continuous pressure and continuous normal particle velocity and by exploiting the symmetry of the problem. Although not limited to scattering from acoustically compact bubble clouds, this problem yields itself to a simple and straightforward analysis using the well established collective oscillations model for the effective sound speed and effective density as outlined in Chapter 2.

In addition to the partial wave expansion, simple analytical expressions were derived which approximate qualitatively, and semi-quantitively, the monopole resonance frequency  $\Omega_0$  (Eq. 3.32) and the TS (Eq. 3.33) at this

resonance of a bubble filled sphere. It was determined that the resonance frequency and TS depend only on two physical characteristics of the bubble cloud: the radius (a) and the void fraction ( $\beta$ ). It is important to stress that the monopole TS level and  $\Omega_0$  can always be obtained without a detailed knowledge of the bubble size distribution.

Since upper ocean dynamics (or even laboratory experiments) will not likely produce spherical bubble clouds, it is obvious that other geometries (cylinders, cones, ellipsoids, or arbitrary shaped volumes etc.) should be explored. As in the solution presented in this chapter, the scattering form function for other geometrical shapes can also be expressed as a series expansion of monopole, dipole, quadrupole and higher order terms. But for long wavelengths (i.e.,  $\lambda > 2\pi a$ ), the monopole (volume) mode is the most important and thus the object can be assumed spherical. Then the scattering can be expressed as a simple function, dependent only on the void fraction and equivalent (spherical) radius. A comparison of a measurement with the model presented here can be made if the total cloud volume and total volume of gas injected into the bubble cloud are known, since the void fraction and equivalent spherical radius are determined from these quantities.

The prime motivation in this effort was to solve the problem of sub-kilohertz backscattering of sound from bubble clouds. At such frequencies, the wave length of sound in water can be considerably greater than the gross dimensions of the cloud. As such, the choice of a sphere is appropriate. Selecting a more complex geometry will not likely enhance the physical understanding of the problem, except of course at higher frequencies. Furthermore, the assump-

tion of uniform void fraction throughout the interior of the sphere, although non–physical from an oceanographic stand point, is nevertheless a first step at understanding the problem.

# Part II

Acoustic scattering from bubble clouds in the absence of boundaries: Experiment

# Chapter 4

# The Lake Seneca experiment: Project overview and bubble cloud generation

### 4.1 Introduction

The Office of Naval Research in the fall of 1988 initiated the Sea Surface Scattering Special Research Program. From this program evolve a small consortium of investigators advocating laboratory and lake experiments designed to determine the mechanisms governing low–frequency sound scattering from bubble clouds. As a result, an experiment was designed to measure the frequency–dependent backscatter from a bubble cloud in fresh water (Lake Seneca, NY) in the absence of boundaries and under known propagation conditions. In these efforts we did not set out to duplicate realistic salt–water

clouds, but rather to obtain data to test and validate a low–frequency bubble cloud scattering theory.

A series of three experiments were performed in the Fall-'90, Summer-'91, and Fall-'91. The experiments were conducted under conditions that were generally calm: daily sound velocity profile measurements were taken which showed isovelocity conditions from a depth of 65 m down to the bottom of the lake. The lake was roughly 160 m deep at the test location, with an additional 120 m of soft mud below this; the distance to the nearest shore was over 1.5 km. The initial experiment, performed at 25 m-depth, successfully served as a qualitative proof of principle that bubble clouds are indeed efficient scatterers of sound at frequencies below individual bubble resonance. However, neither the cloud volume nor the bubble size distribution within the cloud could be measured  $in\ situ$  , and therefore a subsequent modeling effort proved untenable. In addition, the source used to insonate the cloud, a parametric source (PS), relies on a nonlinear interaction of two colinear sound beams in the water to produce a highly directional low frequency sound beam at the difference frequency. Concern existed that the observed scattering returns were due to nonlinear 'enhancement' of the parametric source level caused by the highly nonlinear bubbly fluid, rather than due to scattering from an effective medium possessing low sonic velocity at frequencies below bubble resonance (recall the collective oscillations model). This hypothesis, although deemed unlikely, warranted further exploration using a conventional sound source to substantiate the parametric source measurements. This required that the experiment be performed at greater depth to insure that reverberation from the lake surface did not contaminate the measurements.

To resolve these issues, a second test was performed in which a submersible video camera and both the PS and conventional sources (CS) were
deployed. The camera was mounted on the surface of the bubble maker to
record the size distribution of bubbles ejected from the needle tips and was
also used to image the cloud characteristics (*i.e.*, shape, size, rise velocity,
and void fraction). The conventional sources played an important, but limited, role in the scattering measurements. The objective was to compare the
measurements from the PS and CS in order to belay the notion that parametric enhancement occurred. The advantage of a parametric source is that it
provides a highly directional low frequency beam (at the expense of reduced
source level). This directionality results in reduced volume reverberation from
unwanted scattering sources in the vicinity of the test range<sup>1</sup>. This effort confirmed that a PS is a viable tool to perform directional low frequency scattering
experiments from artificially generated bubble clouds.

The data collected during the second experimental effort verified that excessive nonlinear enhancement did not occur and thus the PS was a viable research tool for performing target strength (TS) level measurements. However, the parametric—source based scattering data was limited to frequencies above 800 Hz due to low signal to noise levels. The third and most important experiment was performed in the Fall of 1991 after reconditioning the PS to enhance its output capability. In this effort, measurable scattering was observed down to frequencies as low as 250 Hz. The data collected during the

<sup>&</sup>lt;sup>1</sup>Fish and possibly other unknown apparatus deployed from the test platform. Ours was not the only experiment being conducted at the time.

final experimental effort clearly indicated that the bubble cloud possessed a broad monopole resonance peak having a TS near -4 dB at frequencies between 300 and 350 Hz. Both the TS and the monopole resonance frequency of the cloud ( $\Omega_0$ ) agreed to within experimental uncertainty with results from the simple analytical model outlined in Chapter 3, given a measured bubble cloud radius of  $a=0.38\,\mathrm{m}$  and void fraction  $\beta=5.4\times10^{-3}$ . Recall that a perfectly reflecting sphere having radius  $a=2\,\mathrm{m}$  possesses a target strength of 0 dB. The fact that a 300 Hz sound wave incident on an acoustically compact bubble cloud ( $a<0.5\,\mathrm{m}$ ) can scatter sound in the free field so effectively is remarkable.

Besides the PS, the most critical piece of equipment employed in these efforts was the National Center for Physical Acoustics (NCPA) bubble cloud generator, for without it this experiment could not have been performed. It consisted of a pressurized steel air reservoir which was vented to the lake by three concentric rings of 22–gauge hypodermic needles (48 needles total). The bubbler could be operated in either a continuous mode in which a steady column of bubbles was produced, or in a transient pulsed mode in which bubble cloud bursts were introduced into the lake on demand. Air was supplied to the bubble maker from a compressor located on the barge and the internal overpressure in the bubble maker and burst duration (the time that air was allowed to flow freely through the needles) were regulated from the surface, allowing bubble clouds to be generated on demand. The operation of the bubble maker is discussed in detail in Section 4.2.

In this chapter, the operation of the transient bubble maker and bubble

cloud characterization measurements are detailed. The total cloud volume, the total gas volume injected into the cloud, the rise velocity, and the bubble size distribution were measured non–acoustically using the video tape data. These parameters are needed to estimate the void fraction and radius of an equivalent volume sphere. Finally, these measurement allow us to make a comparison of the scattering measurements (see Chapter 5) with the simple analytical model presented in Chapters 2 and 3.

#### 4.1.1 Sound speed profile

Shown in Figure 4.1 are measured sound speed profiles obtained during the Summer and Fall-'91 efforts. Profiles were measured daily during the test by lowering a thermistor chain from the edge of the test platform. The near surface sound speed profile (< 10 m) was variable as one might expect. Figure 4.1 clearly illustrates that at depths below 61 m iso-velocity conditions prevailed. This assured us that significant refraction of sound rays propagating across the test range did not occur. Consequently, a ray launched at a declination angle of 35.2° at a depth of 61 m will, in accordance with Snell's law, have a declination angle of 35.2° at a depth of 87.6 m (300 ft). For the calculations presented herein, a sound speed of 1421.5 m/s was used.

# 4.2 Bubble maker deployment and operation

The heart of the experiment was the bubble cloud generator. Illustrated in Figure 4.2, the bubble maker consisted of a submersible pressurized steel air

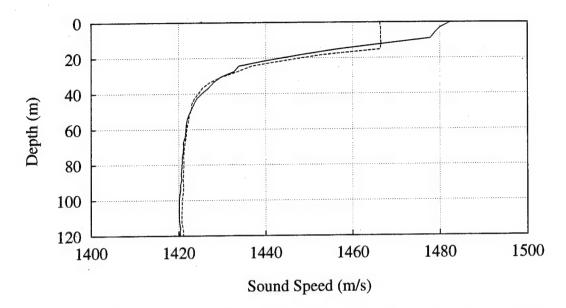


Figure 4.1: Sound speed profile at Lake Seneca test site. The dashed line represents sound speed for the Summer-'91, and the solid line the Fall-'91. At a depth of 87.6 m (300 ft) the sound speed was 1421.5 m/s.

reservoir which was vented to the lake by three concentric rings of 28-gauge hypodermic needles (8, 16, and 24 needles per ring; 48 total). It could be operated in either a continuous mode in which a steady column of bubbles was produced, or in a transient pulsed mode in which artificial cloud bursts were released into the lake. Air was supplied to the bubble maker from a compressor located on the barge. The internal over-pressure in the bubble maker and burst duration (the time that air was allowed to flow freely through the needles) were regulated from the surface, allowing bubble clouds to be generated on demand. A diagram illustrating the bubble maker is shown in Figure 4.3 and its operation is described below.

The bubble maker was lowered over the edge of the barge to a depth of 91.4 m beneath the lake surface, or 3.8 m beneath the expected backscat-

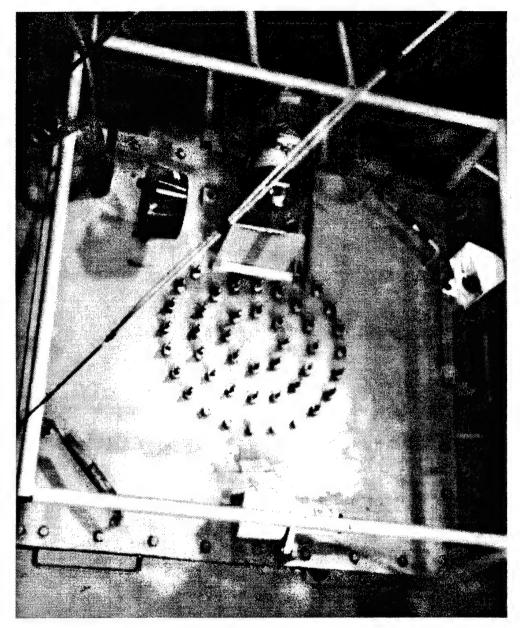


Figure 4.2: Photograph of the bubble maker.

ter target position (see Figure 5.1 for the layout of the test range) . It was supported by a davit mounted on the top face of the barge with a 1/4-inch steel cable. The depth of the bubble maker was determined using the cable

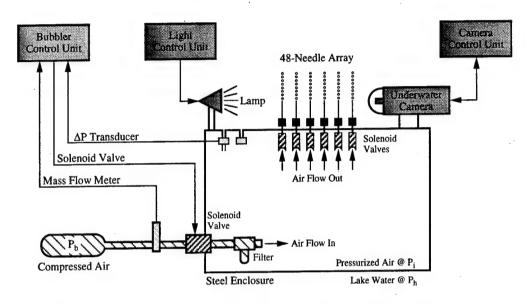


Figure 4.3: Schematic of the Seneca Lake bubble maker.

meter used to deploy the hydrophones and the steel target sphere. In addition, an absolute electronic pressure transducer was mounted on the interior of the bubbler, but exposed to its exterior. This provided a secondary means of determining the depth by measuring the ambient hydrostatic pressure. Electrical power was provided to this pressure transducer and the other electronic elements (solenoid valves, and the  $\Delta P$  transducer) via a 10 conductor submersible cable connected to the Bubbler Control Unit (affectionately named  $Deep\ Thought$ ) located inside the barge.

Since the hydrostatic pressure increases with depth, the bubbler depth and internal pressure were increased incrementally, to prevent implosion. At each 5 m depth increment the lowering process was halted and several clouds were released from the bubbler in order to purge water from the syringe nee-

dles and to insure re-equilibration of the internal pressure to a level 10 psi greater than the hydrostatic pressure at depth. Individual cloud releases were visually monitored with the PhotoSea Cobra submersible video camera which was mounted on the surface of the bubble maker (see Figures 4.2 and 4.3) in order to confirm that air flowed freely through all the needles.

During the deployment phase, the internal pressure of the bubble maker was maintained at 10 psi above the ambient hydrostatic pressure at a given depth. To consistently and accurately maintain the over–pressure inside the bubbler an electronic "regulator" system was designed which consisted of the following components: an electric differential pressure transducer ( $\Delta P$ ); an electronic input solenoid valve (normally closed); an air supply line pressurized 30 psi over the ambient pressure at depth and connected to the input solenoid; and the bubble maker controller, *Deep Thought*.

The  $\triangle P$  transducer measured the difference in the external hydrostatic pressure relative to the interior of the bubble maker, transmitting it as a voltage to  $Deep\ Thought$ . The overpressure was configured automatically controlled and nominally set to 10 psi maximum and 9.5 psi minimum. Throughout the experiment,  $Deep\ Thought$  continuously monitored the  $\triangle P$  to assure that the over–pressure did not drop below the preset value of 9.5 psi. As a precautionary measure, an analog pressure gauge displaying the difference in internal and external pressure of the bubble maker was installed on the surface of the bubble maker and was occasionally monitored using the underwater video camera. Data from this pressure gauge was not recorded since the electronic transducers performed as expected.

When  $\triangle P$  fell below 9.5 psi, the bubble maker controller activated the normally-closed "input solenoid" mounted inside the bubbler allowing air to flow into the reservoir. Once the input solenoid was activated, the inflow of air continued until  $\triangle P$  reached 10 psi. The pressurized air was provided by a compressor located on the barge surface; as a safety backup, a compressed nitrogen cylinder was also placed inline. The air source was pressurized to 30 psi above the internal pressure of the bubbler in order to fill it quickly as clouds were produced. In addition, air filters (designed trap particulates and fluids) were placed inline at the compressor output and the bubble maker input.

Three concentric rings containing 8, 16, and 24 needles (22–gauge) were mounted on the surface of the bubble maker (see Figure 4.2). Each pair of needles, opposite one another on a given ring, were controlled via a single normally-closed solenoid valve mounted in the interior of the bubble maker. The burst duration, the time that air was allowed to freely flow from the needle tips, and the opening of a single, pair, or all rings of needles were controlled remotely from *Deep Thought*. The cloud production parameters are summarized in Table 4.1.

	# Needles = 48
$t_{\rm burst} = 3.4{\rm s}$	$Depth = 91.4 \mathrm{m}$

Table 4.1: Cloud generation parameters.

In Figure 4.4, the digitized and scaled output from the  $\triangle P$  transducer is displayed. A single cloud was released at t=0 s at which point the internal over–pressure was 9.9 psi. As air exited the bubble maker, the internal pressure

decreased. When the internal pressure reached 9.5 psi, the air supply input solenoid valve was opened forcing air into the bubble maker; a corresponding increase in the over–pressure is observed. To end bubble cloud production (at time  $t_{\rm burst}=3.4\,\rm s$ ), the solenoid bank controlling the needle ports was deactivated. Air from the supply line continued to fill the bubbler and  $\Delta P$  quickly returned to 10 psi, at which point the input solenoid was closed. Note that in Figure 4.4, the over–pressure decreases slightly during the next 40 s after the bubble cloud generation. This was casued by a pair of needles which continuously released a few bubbles per second as viewed on the video monitor. This leak was small and of negligible importance.

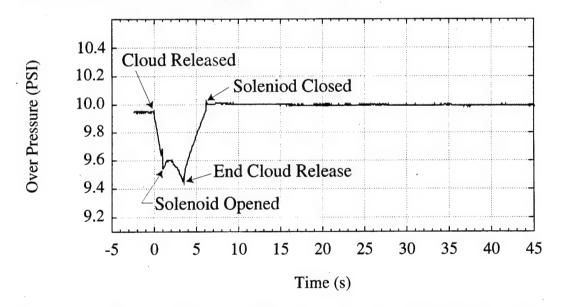


Figure 4.4: Pressure difference ( $\triangle P$ ) between the lake and the bubble maker at a depth of 91.6 m as a function of time during a cloud release. The cloud was released at t=0 at which point the internal pressure decreased. The electronic regulator was set to a 0.5 psi threshold at which point the input solenoid was opened allowing air to enter the bubble maker.

# 4.3 Bubble cloud characterization

In order to make a comparison between the transient bubble cloud scattering measurements and the model, it is necessary to have a detailed knowledge of the cloud shape and volume, bubble size distribution, rise velocity, and position. Since our interest is in backscattering, these values need only be known when the bubble cloud passes through the backscatter target position (recall Figure 5.1). With this information, the effective phase speed of sound in the bubbly mixture can be estimated. Furthermore, by solving the boundary value problem of the system, the backscatter TS as a function of frequency can be determined. The solution of this problem will result in the resonance frequencies or modes of the system, as well as the scattering cross section (or target strength) as a function of frequency. However, our ability to measure the physical attributes of the bubble cloud (especially the shape and spatial distribution of bubbles) with accuracy was limited by various physical constraints. Therefore explicit agreement of the scattering model presented in Chapters 2 and 3 with the measurements (Chapter 5) is difficult, except at the lowest frequencies (i.e., ka < 1).

For low frequencies, where the bubble cloud is considered acoustically compact, the shape is of secondary concern provided the aspect ratio does not vary appreciably from unity. Recall that the lowest order mode of an acoustically compact scatterer is dependent on the volume of the target and thus the spherical model outlined earlier should be sufficient to predict the monopole scattering characteristics of the bubble cloud. To estimate the TS of a bubble cloud in the free-field at the lower frequencies, only the volume

fraction ( $\beta$ ) and cloud radius (a) need to be known. These two quantities were obtained by measuring the total cloud volume ( $V_{\text{cloud}}$ ) and the total volume of air ( $V_{\text{gas}}$ ) injected into the cloud, since  $\beta = V_{\text{gas}}/V_{\text{cloud}}$  and  $V_{\text{cloud}} = 4\pi a^3/3$ . For an acoustically compact bubble cloud, the bubble size and spatial distribution is of minimal importance as long as the frequency of the incident wave is much less than the resonance frequencies of the constituent bubbles (*i.e.*,  $kR_o \ll ka \ll 1$ ). To verify that the prior condition was satisfied, the bubble radius distribution was also measured as well.

#### 4.3.1 Bubble cloud rise time

The rising bubble clouds viewed on the monitor at a height of 3.8 m above the bubble maker appeared to be nominally cylindrical shaped with a blunt leading edge and a trailing skirt of bubbles. The distinction between the 'skirt' and the more densely populated regions of the bubble cloud was not obvious and therefore an "educated-guess" was made. The video camera was deployed in the configuration shown in Figure 4.5. It was not possible to view an entire bubble cloud at any instant in time. This was due to the short separation distance between the bubble cloud and the submersible camera (< 0.6 m). These measurements were performed on the last scheduled day of the experiment, preventing redeployment of the camera and lighting rig.

To determine the rise velocity, shape, and total volume of the bubble cloud as it crossed the axis of the test range, video recordings were made. During these measurements, the Photosea Cobra underwater video camera and lighting system were mounted at the backscatter target position on the

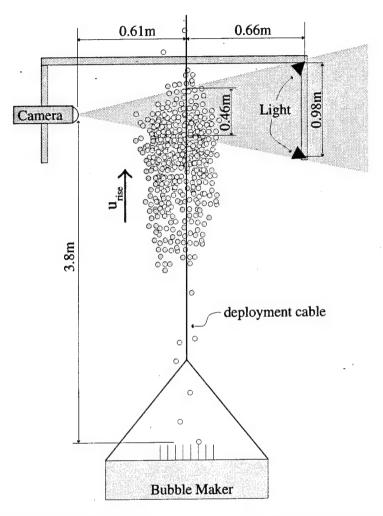


Figure 4.5: Video rig used to measure physical parameters of cloud at depth.

suspension cable 3.8 m above the bubble maker. A diagram illustrating the geometry of the camera and lighting station used to perform the rise time, and cloud volume and shape measurements is shown in Figure 4.5. The video camera was focussed on the plane perpendicular to the suspension cable and the field of view was approximately 0.46 m vertical and horizontal in the focal plane. The viewing area is indicated in the Figure 4.5 by the triangular region.

The bubble cloud rise–time into the backscatter position was measured

from a series of 30 bubble cloud releases separated by 10 s intervals. From the rise-time measurements, the cloud rise velocity and cloud length were inferred. The rise-times for the upper and lower cloud boundaries to pass the video camera relative to the beginning of cloud release (t=0) were measured using a digital stop watch. The top rise time was  $t_{\text{top}}=9.1\pm0.5\,\text{s}$  and the bottom rise time was  $t_{\text{bottom}}=13.6\pm0.5\,\text{s}$ , where the error indicates standard deviation in the measurements.

#### 4.3.2 Bubble cloud rise velocity

The velocity of the upper and lower boundaries of the bubble cloud were deduced by dividing the rise distance ( $z = 3.8 \,\mathrm{m}$ ) by the appropriate rise times as follows:  $u_{\rm top} = z/t_{\rm top}$ , and  $u_{\rm bottom} = z/t_{\rm bottom} - t_{\rm burst}$ , where  $t_{\rm burst} = 3.4 \,\mathrm{s}$  is the burst duration (the length of time that air was allowed to escape from the bubble maker needle tips). Consequently,  $u_{\rm top} = 0.41 \pm 0.034 \,\mathrm{m/s}$  and  $u_{\rm bottom} = 0.37 \pm 0.012 \,\mathrm{m/s}$ , where the error is estimated using a modified Taylor expansion as described in Appendix D.

The mean velocity of the bubble cloud,  $u_{\rm rise} = 0.39 \pm 0.023 \,\rm m/s$ , is the average of  $u_{\rm top}$  and  $u_{\rm bottom}$ . It should be noted that the cloud boundaries—top, radial, and especially bottom—are not well defined and thus an "educated" guess was made.

### 4.3.3 Bubble cloud length

The cloud length is given by the simple expression:  $L = u_{\text{rise}} (t_{\text{bottom}} - t_{\text{top}})$ . For the rise times and cloud velocity given above, the cloud length is:  $L = u_{\text{rise}}$ 

# 4.3.4 Bubble cloud shape and volume

In addition to the difficulty of accurately determining the upper and lower boundaries of the bubble cloud, the sides of the bubble cloud was not well defined. Furthermore, the width of the rising bubble cloud exceeded the field of the view on the video monitor because of the inadequate spacing between the video camera and the rising bubble cloud (see Figure 4.5). To compensate for this fact, the video camera was panned to the left of the cloud in order to record an image of the right edge of the several rising clouds; similarly, the camera was panned to the right to view an image the left half of several rising clouds. In both cases, the cable used to lower the bubbler and to support the video camera and lighting rig was kept in the field of view and used as a fixed point of reference.

To determine the cloud shape and volume, the video tapes were played back and the distance between the reference point (the support cable) and a edge of the bubble cloud was measured as a function of cloud height (actually rise time). Since the entire cloud did not fit into the field of view, a sequence of photographs were taken directly from a video monitor. The 35 mm camera used to take these snapshots was mounted on a tripod and focussed on the video monitor. During the playback of a rising bubble cloud, the video tape was paused when upper boundary of the cloud was near the top of the field of view and a single photograph taken. Next, the video tape was advanced 1.4s (i.e., the amount of time required for a portion of the cloud visible on

the lower part of the video monitor to rise above the field of view), paused, and another snapshot taken. This process was repeated until an entire bubble cloud had passed through the viewing area.

A total of seven photographic sequences of rising bubble clouds were obtained (four right-side images, and three left-side images). A pair of these sequences, shown in Figure 4.6, were assembled and roughly illustrate the shape of a single rising bubble cloud.

It is clear from Figure 4.6 that the spatial distribution of bubbles within the cloud is not uniform, an assessment that is confused by the non-uniform lighting used to illuminate the cloud. Thus, an attempt to estimate the spatial bubble population density within the cloud was not made. In Figure 4.6, the curvature or "bulges" indicated in each photograph of this mosaic is due to distortion of the image by the video camera lens or its perspex pressure dome. When viewed on the video monitor, the cloud appeared to "roll" rather than rise.

The bubble cloud image in Figure 4.6 was divided into 32 segments equal segments about the cloud length ( $dL \approx 5\,\mathrm{cm}$ ) and the radial versus height profiles of the left and right hand edge of several clouds were estimated relative to the suspension cable. This radius—height profile was then used to determine the total volume of the bubble cloud by accumulating the volumes of conical frustum bounded by adjacent radii and height dL. A sample frustum is illustrated in Figure 4.7, and the total cloud volume is given by

$$V_{\text{cloud}} = \frac{\pi}{3} \sum_{i=1}^{32} (a_i^2 + a_i a_{i+1} + a_{i+1}^2) dL,$$

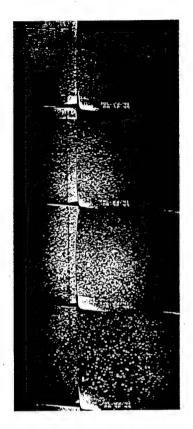


Figure 4.6: Mosaic of a bubble cloud. A series of snapshots were taken directly from the video monitor during tape playback. The images to the left and to the right were not taken from the same cloud. However, the sequence of photos in each column were obtained from a single rising bubble cloud. In each column, each consecutive snapshot is separated in time by approximately 1.4 s. Notice the variation in bubble population density. The dark vertical stripe in the left portion of the image is the suspension cable used to lower the bubble maker. Thus it is clear that the bubble cloud rose slightly off axis due to a slight current in the lake.

where  $a_i$  refers to the *i*-th radial element. The total cloud volume then is  $V_{\rm cloud} = 222 \pm 0.027 \, l$ , where the radius-height profile is given in Table 4.2.

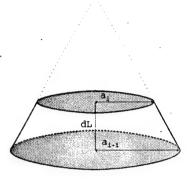


Figure 4.7: Frustum of a cone. The total volume of the bubble cloud was determined by accumulating 31 frustum volumes using the radial measurements in Table 4.2

#### 4.3.5 Effective bubble cloud radius

Although the bubble cloud image in Figure 4.6 is clearly non–spherical, the monopole scattering response (volume mode) can be estimated using the sphere model outlined in Chapter 3. Thus for a spherical cloud of volume  $V_{\text{cloud}} = 222 \pm 0.027 \, l$ , the radius of the cloud is  $a = 0.37 \pm 0.014 \, \text{m}$ . We are now left to determine the net void fraction of the bubble cloud.

## 4.3.6 Total gas volume measurement

To determine the total volume of gas injected into a single bubble cloud, an precision electronic flow meter (Porter Instrument Co., Model 113) was mounted inline with the bubble maker air supply line on the barge surface.  $V_{\rm gas}$  is a critical quantity needed to determine the void fraction, and thus used to make an unbiased comparison of the model with the acoustic scattering measurements. The voltage output from the flow meter is proportional to the

Table 4.2: Radius versus height profile of the Lake Seneca bubble cloud.

Height	Radius	Volume
(m)	(m)	(l)
0.00	0.106	2.150
0.05	0.121	2.506
0.11	0.124	3.044
0.16	0.146	3.949
0.21	0.162	4.755
0.27	0.176	5.391
0.32	0.183	5.777
0.37	0.188	5.961
0.43	0.189	6.227
0.48	0.197	6.725
0.53	0.204	7.113
0.58	0.208	7.436
0.64	0.214	7.777
0.69	0.218	8.058
0.74	0.221	8.437
0.80	0.228	8.931

Height	Radius	Volume
(m)	(m)	(l)
0.85	0.234	8.6
0.96	0.238	9.3
1.0	0.24	9.2
1.1	0.23	9.1
1.1	0.23	9.2
1.2	0.23	9.0
1.2	0.23	8.6
1.3	0.22	8.2
1.3	0.22	7.9
1.4	0.21	7.6
1.4	0.21	7.2
1.5	0.20	6.8
1.5	0.20	6.2
1.6	0.19	5.9
1.7	0.19	6.0
1.7	0.19	

Note: In the middle columns, the cloud radius as a function of height is given. In the third columns, the volume element (in liters) of a slice of a right circular cone is given for consecutive radial pairs. The total cloud volume is obtained by summing the volume elements (see Figure 4.7).

mass flow of gas in standard liters<sup>2</sup> through the apparatus. During a cloud release, the voltage output from the flowmeter was digitized and integrated with respect to time, and scaled to yield the total volume of gas in standard liters  $(V_{\rm std})$ . Then  $V_{\rm gas} = (V_{\rm std}/P_h) \cdot 10^5 \, {\rm Pa}$ , by the ideal gas law. Here,  $P_h = 9.82 \times 10^5 \, {\rm Pa}$  is the ambient hydrostatic pressure at a depth of 87.6 m beneath the lake surface.

As a bubble cloud was released from the bubble maker,  $\triangle P$  immediately

<sup>&</sup>lt;sup>2</sup>One standard liter is equivalent to one liter at atmospheric pressure (i.e., , re 10<sup>5</sup> Pa)

decreased (recall Figure 4.4). When  $\triangle P$  reached 9.5 psi, the input solenoid was opened allowing air to flow into the bubble maker. The air flow entering the bubble maker, as illustrated in Figure 4.8, increased rapidly after the input solenoid was opened ( $t=1.2\,\mathrm{s}$ ), leveled off ( $t=3.4\,\mathrm{s}$ ), and decreased after the bubble maker reached its equilibrium internal over–pressure when the input solenoid was closed ( $t=6\,\mathrm{s}$ ).

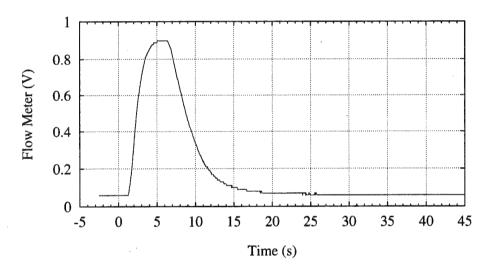


Figure 4.8: Digitized flow meter output as a function of time during a cloud release. The output was integrated and scaled to yield the total air volume injected into the cloud, nominally  $1.22 \pm .014 l$ .

During the Seneca-2 experiment, the mass flowmeter output from total of 112 cloud releases was digitized at 10 points per second. The data from each cloud release was integrated and scaled to yield the cumulative mass flow in standard liters. This quantity was then converted to the equivalent gas volume at a depth of 87.6 m beneath the lake surface (the expected backscatter position of the bubble cloud). Over the course of the experiment, and for the cloud generation parameters listed in Table 4.1, the measured total gas volume

injected into the Lake Seneca bubble cloud was  $V_{\rm gas} = 1.22 \pm .014 \, l$ , where the error is the standard deviation of the measurements (recall that the total cloud volume was  $V_{\rm cloud} = 222 \pm 0.027 \, l$ ). Thus, the mean void fraction for a single Lake Seneca bubble cloud was  $\beta = 0.0055 \pm 0.001$ .

As discussed in Chapter 2, the sound velocity in the bubbly mixture is well approximated by the Wood expression (Eq. 2.9) and is only dependent on the void fraction when the driving frequency of the acoustic wave is well below the fundamental resonance frequencies of the constituent bubbles. However, for large ambient pressures, like that experienced by the bubbles during the this experiment, the mixture sound speed is known to diverge from the low frequency limiting expression (Commander & Prosperetti, 1989). After simple algebraic manipulation and an order of magnitude analysis, it is clear that an improved approximation to the Wood equation can be derived from Eq. (2.24)

$$c_e \to \left(\frac{\kappa P_{\infty}}{\frac{c^2}{\kappa P_{\infty}} + (1 - \beta)\beta\rho}\right)^{\frac{1}{2}}.$$
 (4.1)

Eq. (4.1) is termed the modified Wood expression.

Using the estimated values of the void fraction and cloud radius and applying Eq. (4.1), (3.32), and (3.33), the low frequency phase speed  $(c_e)$ , monopole resonance frequency  $(\Omega_0)$  and backscatter TS of the Lake Seneca bubble cloud can be estimated—assuming a spherical cloud. A further assumption made here is that the resonance frequencies of all the bubbles contained in the Seneca cloud are much greater than  $\Omega_0$  (this will be discussed in the next section).

#### 4.3.7 Bubble radius distribution measurements

With the video camera and lighting rig deployed 3.8 m above the bubble maker as in Figure 4.5, it was not possible to accurately measure the bubble size distribution in situ. However, with the video camera mounted on the top surface of the bubble maker (as shown in Figure 4.2), imaging the size distribution of bubbles produced at a single needle tip was possible.

The Photosea Cobra video camera could be focussed down to very short distances (even onto its own perspex pressure dome), enabling us to place it very close to a single needle tip and observe single bubbles as they were released. In addition, the camera was shuttered so that the rising bubbles were not blurred depreciably, and we could therefore make precise BSD measurements. A frosted semi-opaque screen was mounted a few cm behind one of the needles and used to diffuse the background lighting from the 500 Watt Halogen bulbs. On the diffuser, a millimeter scaled ruler was secured as a second reference calibration for the measurements. In addition, the diameter of the 22 gauge needle, 0.7 mm, was used as a reference measurement. In Figure 4.9 a digitized image as viewed from the submersible video camera mounted on the surface of the bubble maker and focused on a single needle tip is shown. As was generally the case, only a few bubbles per image were in focus due to the narrow field of view. The internal over-pressure in the bubble maker was set to 10 psi and the bubble maker was pulsed for 3.4s to generate a cloud.

Images of newly formed and rising bubbles were recorded on video tape.

The size distribution of in–focus bubbles emerging from the needle tip were then measured directly from the surface of a video monitor during tape play-

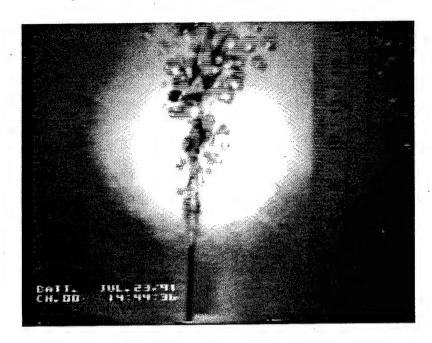


Figure 4.9: Bubbles released from single needle tip as viewed from the submersible video camera. The scaled ruler (mm gradations) is visible on the right. The bright spot in the center is due to the intense lighting needed to illuminate the rising bubbles.

back. A digitized video image is shown in Figure 4.9. Here, it is clear that the newly formed bubbles had not yet reached an equilibrium shape, and many of the bubbles are blurred due to the narrow focal plane. Because the majority of the bubbles were not spherical, it was essential to measure both the major and minor axis of the bubbles. The bubble volume was estimated using simple geometric formulas for the volume of oblate and prolate spheroids:

$$egin{array}{lll} V_{
m oblate} &=& rac{4\pi}{3}a^2b, \ V_{
m prolate} &=& rac{4\pi}{3}ab^2, \end{array}$$

where, a and b are the semi-minor and semi-major axis respectively.

It is likely that the observed spheroidally shaped bubbles were prolate (i.e., link sausages), rather than oblate (i.e., pancakes), since they were forcefully ejected from the needle tip and had not yet obtained an equilibrium shape. Once the bubble volume was determined, the equivalent spherical bubble radius and corresponding statistical distribution were calculated and are illustrated in Figure 4.10 and listed in Table 4.3.

Since the radius of these bubbles are in the range 0.42 mm to 2.93 mm, then at a depth of roughly 91 m the range of bubble resonance frequencies is bounded roughly by 2.9 kHz and 20 kHz, according to the Minneart resonance frequency (Eq. 2.21). Thus, for all acoustic frequencies below about 2.0 kHz, the speed of sound in a cloud composed of these bubbles should be well approximated by the modified Wood expression (Eq. 4.1).

Mean	$1.55\mathrm{mm}$	Min	$0.42\mathrm{mm}$
Std. Dev.	$1.40\mathrm{mm}$	Max	$2.92\mathrm{mm}$
Counts	159	Range	$2.5\mathrm{mm}$

**Table 4.3:** Statistics from *in situ* BSD measured slightly above a single needle tip at Lake Seneca.

In Figure 4.10, there are clearly three visible peaks located at  $R_{o,1} = 0.39 \,\mathrm{mm}$ ,  $R_{o,2} = 1.45 \,\mathrm{mm}$ , and  $R_{o,3} = 1.95 \,\mathrm{mm}$ . This BSD data was fitted to a triple–Gaussian curve

$$f(a) = A_1 e^{-\left(\frac{(a - R_{o,1})}{\sigma_1}\right)^2} + A_2 e^{-\left(\frac{(a - R_{o,2})}{\sigma_2}\right)^2} + A_3 e^{-\left(\frac{(a - R_{o,3})}{\sigma_3}\right)^2}$$
(4.2)

using a nonlinear least-squares method. The parameters obtained by the fit are given in Table 4.4.

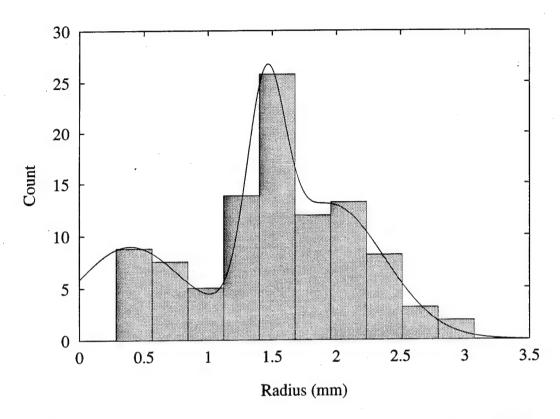


Figure 4.10: Bubble size distribution measured just above a single needle tip at Lake Seneca. The curve represents a triple-gaussian non-linear least squares fit to the measured data. For phase speed calculations the fit is truncated between 0.1 mm and 3.2 mm.

	Mean	Std. Dev.	Amp.
	(mm)	(mm)	
Peak 1	0.391946	0.59941638	8.962
Peak 2	1.44917	0.2074013	19.679
Peak 3	1.947760	0.60210049	12.982

**Table 4.4:** Parameters obtained from a nonlinear least–squares fit of the *in situ* BSD to a triple–Gaussian curve.

In Fig. 4.11, the frequency dependent phase speed versus frequency curve for the Lake Seneca bubble cloud is shown for each of the measured bubble size distributions (*i.e.*, monodispersed, discrete, and Gaussian). For the

monodispersed distribution, the equilibrium bubble radius is  $R_o = 1.55 \, \mathrm{mm}$ , the volume fraction is  $\beta = 0.54 \times 10^{-2}$ , and the ambient pressure is  $P_\infty = 9.82 \times 10^5 \, \mathrm{Pa}$ . For the discrete distribution, the mixture sound speed is calculated using the radius and amplitude values shown in the histogram of Figure 4.10. Finally for the Gaussian fitted bubble radius distribution, the triple Gaussian fit (Table 4.4) to the histogram in Figure 4.10 is applied. Note that in all cases, the phase speed is independent of frequency well below the bubble resonance frequency. In the low frequency limit the calculation of the mixture sound speed yields  $c_e = 464 \, \mathrm{m/s}$ . This compares favorably with the modified Wood expression (Eq. 4.1) value of 468 m/s in the adiabatic limit ( $\kappa \to \gamma$ ). In the isothermal limit ( $\kappa \to 1$ ), the modified Wood expression predicts a sound speed of 402 m/s.

## 4.4 Summary

Artificial transient bubble clouds were generated at a depth of 87.6 m beneath the surface of a freshwater lake for the purpose of conducting an acoustic backscattering experiment. Although the bubble cloud image in Figure 4.6 is clearly non–spherical, the monopole scattering response (volume mode) can be estimated using the sphere model outlined in Chapter 3. Thus for a spherical cloud of volume  $V_{\rm cloud} = 222 \pm 0.027 \, l$ , the radius of the cloud is  $a = 0.376 \pm 0.014 \, {\rm m}$ . In Table 4.5, the values obtained during the processes described herein are summarized.

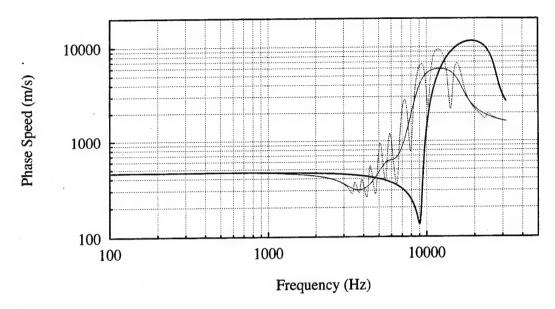


Figure 4.11: Effective phase speed versus frequency for the Lake Seneca bubble cloud using monodispersed (solid line), discrete (jagged–dashed line), and Gaussian (smooth–dashed line) bubble size distributions. Here the void fraction is  $\beta = 0.55 \times 10^{-2}$ , and  $P_{\infty} = 9.82 \times 10^{5} \, \mathrm{Pa}$ . For all frequencies of interest in this experimental analysis (*i.e.*, below 2000 Hz, the sound velocity in the mixture asymptotes to 460 m/s.

Table 4.5: Lake Seneca experimental precision analysis.

#### (a) Measurement

				Relative
		Mean		Uncertainty
Para	ameter	Value	Uncertainty	(%)
$P_o$	(Pa)	958774	15336	1.60
$V_{ m cloud}$	(1)	222	2.7	12.28
$V_{\rm gas}$	(l)	1.2	0.1	11.48
c	(m/s)	1421.5	0.000	0.00
$\overline{a}$	(m)	0.376	0.014	3.72
β		0.55	0.001	16.80

#### (b) Predictions

$\rho_{\epsilon} = (\mathrm{kg}/m^3)$	994.73	0.928	0.09
$c_e \text{ (m/s)}$	402.05	31.049	7.72
$\Omega_0$ (Hz)	294.088	25.213	8.57
$k_{o}a$	0.489	0.046	9.35
TS (dB)	-2.998		

# Chapter 5

# The Lake Seneca experiment: acoustic backscattering measurements and analysis

# 5.1 The experimental test plan

The Lake Seneca facility consisted of two moored barges in a water depth of 130 m, with the smaller barge (called the TCP barge; 10.7 m x 42.7 m) serving as the platform for the test range. Equipment was deployed with the use of davits, a cable meter, and the edge of the barge, with the resulting vertical geometry shown in Figure 5.1. Transmitters and receivers were oriented colinearly, with the axis of the range intersecting the path of a rising bubble cloud. A depth of 87.6 m was chosen for the target backscatter position in order to minimize both surface reverberation and volume reverberation (i.e.,

from fish, etc.) in the vicinity of the staging. Because of the deployment depth and the extent of the test range, the lake surface did not play a role in the backscattering and thus a comparison of the scattering measurements with the free-field model presented in Chapter 3 is facilitated.

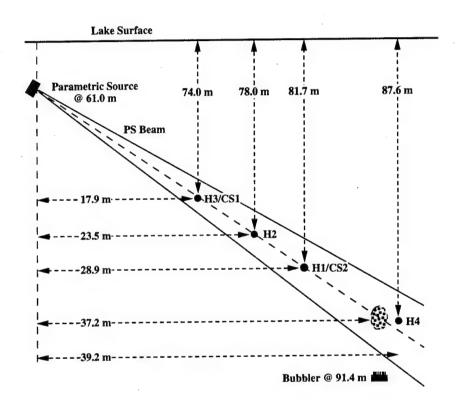


Figure 5.1: Layout of the test range. The receivers in the backscatter direction are denoted by H1, H2, and H3. Hydrophone H4 is in the forward direction and was used primarily to measure noise emissions from newly formed bubble clouds (Nicholas et al., to be submitted). H3 and H1 were also used as a conventional sources (CS1 and CS2, respectively) during some of the tests. The PS beamwidth is indicated approximately by the region between the solid lines.

The backscatter target strength of the rising bubble cloud was measured over frequencies ranging from  $250\,\mathrm{Hz}$  to  $14\,\mathrm{kHz}$  using both conventional

and parametric sources. Over the course of the measurements, the source repetition frequency was fixed at either 1 or 2 Hz and the incident pulse length made to vary from 8 to 15 ms. Calibrations of both the frequency–dependent source level and beamwidth were performed in situ and were nominally 177 dB (re: 1  $\mu$ Pa at 1 m) and 8.0°, respectively (at 1.0 kHz). In addition, H1/CS2 and H3/CS1 were used as conventional sources in some of the tests.

#### 5.1.1 The parametric source

Accurate target strength measurements are difficult to acquire as the frequency of the source is lowered, primarily because of the difficulty of radiating sound with narrow beam widths in order to avoid boundary reflections and excess volume reverberation within the sea. Conventional acoustic projectors (i.e., piston sources) capable of the necessary beam widths (< 10°) are both cumbersome and expensive. Furthermore, a conventional source radiates a significant amount of energy into the sidelobes. For a conventional projector, a 10-m diameter transducer aperture would be required to produce a 10°-3 dB¹ beam width at 1000 Hz (Konrad & Geary, 1991).

With a parametric source, a narrow beam with negligible side lobes is produced from a manageable sized projector. A 10° beam width at 1000 Hz can be achieved using a 0.75 m diameter aperture. In addition, the same source can be effectively used at frequencies between 500 and 7000 Hz. The PS produces these unique characteristics by simultaneously radiating sound at two much higher frequencies (called primaries) in collinear beams, and capitaliz-

<sup>&</sup>lt;sup>1</sup>A -3 dB beam width refers to the half power points on the main lobe of the beam.

ing on the inherent nonlinearity of the water to convert some of the primary energy into sound at the difference frequency of the two primary waves. The resulting sound beam is a combination of the large amplitude primary waves and the somewhat lower amplitude (about 40 dB down) "sum" and "difference" frequencies. Acoustical mixing occurs only within the overlap zone of the primary beams, and thus the difference frequency is generated in a long virtual end–fire array extending outward from the source. The fundamental disadvantage of this source is its low efficiency, typically less than 5%, and consequent need for relatively high input powers. However, in the generation of short pulses on the order 100 ms and for duty cycles under 5%, the energy requirements are relatively easy to meet.

The parametric source used in this experiment was Section–4 of NUWC's² (formerly NUSC³) TOwed Parametric Source (TOPS), which is well characterized in Scientific and Engineering Studies: Nonlinear Acoustics 1954 to 1983. The source was deployed to its maximum depth of 61 m (200 ft), using sections of 6" diameter steel pipe. The fundamental design frequency of TOPS was 22 kHz. The two primary frequencies are symmetrically up and down shifted relative to 22 kHz. The signals were time gated and amplified by a 20 kW drive amplifier. Thus, to deliver a difference frequency  $f_{\Delta} = 500 \, \text{Hz}$  into the test range, the source was simultaneously driven at primary frequencies of  $f_1 = 21.75 \, \text{kHz}$  and  $f_2 = 22.25 \, \text{kHz}$ . The source is compact, quite directional, and has a negligible side lobe structure compared to a CS. To perform the measurements described herein at difference frequencies less than

<sup>&</sup>lt;sup>2</sup>Naval Undersea Warfare Center

<sup>&</sup>lt;sup>3</sup>Naval Underwater Systems Center

 $2\,\mathrm{kHz}$ , a low–pass filter (2 kHz) was inserted at the output of each hydrophone in order to remove the primary source frequencies. A measured beam pattern for the PS driven at  $f_{\triangle} = 500\,\mathrm{Hz}$  is illustrated in Figure 5.2. The beam pattern was obtained at receiver H1 (recall Figure 5.1) by tilting the PS about the polar (vertical) axis. In Table 5.1, the -3 dB beamwidth is listed for frequencies between 500 and 4000 Hz.

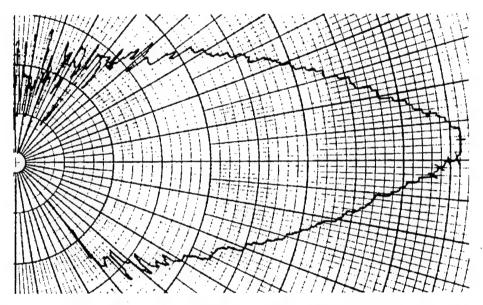


Figure 5.2: Parametric source beam pattern obtained for  $f_{\triangle} = 500 \, \text{Hz}$ . The source was tilted about its vertical axis and the resulting beam pattern measured at receiver H1. The smallest angular and radial units on the grid are separated by 1° and 1 dB respectively. Thus, the  $-3 \, \text{dB}$  beamwidth at 500 Hz is  $\pm 4.25^{\circ}$  or  $8.5^{\circ}$  total.

### 5.1.2 Bubble cloud generation

The heart of this experiment was the bubble cloud generator (Figure 4.3). Air from a steel enclosure, pressurized to  $\triangle P = 10 \, \mathrm{psi}$  above the ambient pressure in the lake (i.e.,  $\triangle P = P_{\mathrm{internal}} - P_{\mathrm{hydrostatic}}$ ), was vented into the lake through

Fq (Hz)	SL (dB)	BW (-3 dB)
500	167	8.5°
1000	177	8.0°
2000	185	7.0°
4000	193	6.5°

Table 5.1: Frequency dependent source level and beam width for the parametric source. These measurements were performed by tilting the PS about the vertical and using receiver H1.

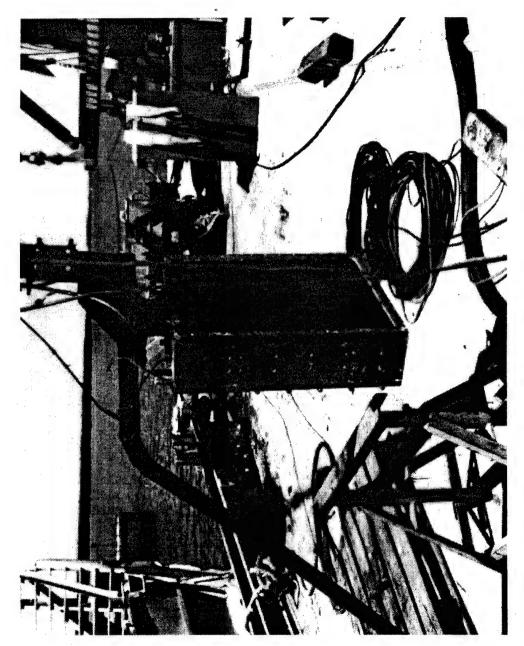


Figure 5.3: Photograph of the parametric source. The projector surface is approximately 0.75 m square.

an array of ports opened via solenoid valves controlled from the surface (see Section 4.2 for a more comprehensive discussion). Each port was capped with a 22-gauge hypodermic needle, the end of which was cut perpendicular to the needle shaft ("pipetting needles"). In all there were 48 ports arranged in a circular array of three concentric rings of radii 6.4 cm (containing 8 ports), 12.8 cm (containing 16 ports), and 19.2 cm (containing 24 ports).

Pressurized air was supplied to the steel enclosure from a compressor located at the surface. After suitable pressure regulation and filtering, the total mass of air supplied to the apparatus was recorded using a precision mass flowmeter. The relative pressure between the air in the enclosure and the ambient lake pressure  $(\triangle P)$  was continually monitored using an electronic differential pressure transducer; during the venting process, if  $\triangle P$  dropped below a preset tolerance, a solenoid valve was opened which enabled compressed air from the supply to enter the pressurized steel enclosure and thus to reestablish the operating overpressure. The regulated air pressure at the compressor was typically 20 psi greater than the internal pressure in the bubble maker.

Bubble clouds were generated in the following manner: At the surface, the overpressure and burst duration (the time that the solenoid bank was opened) were configured to 10 psi and 3.4 s respectively. Clouds were released on demand (by pressing a button), resulting in the formation of rising bubble cloud. The bubble clouds were nominally cylindrically shaped with at blunt leading edge and a trailing skirt of bubbles. The rise velocity of the cloud was approximately  $0.4 \,\mathrm{m/s}$ , and the total cloud volume and total gas volume contained within the cloud were  $V_{\rm cloud} = 222l$  and  $V_{\rm gas} = 1.22l$ , respectively.

The total cloud volume and rise velocity were determined from video recorded data, and the total gas volume from the electronic precision mass flow meter. Although the spatial distribution of bubbles in the cloud was not uniform (recall Figure 4.6), the average void fraction was determined to be  $\beta = 0.0055$  and the equivalent volume spherical radius was a = 0.376 m.

## 5.2 The measurement of target strength

In Appendix A the standard definition of the backscatter target strength is given and the process of performing an experimental measurement is described. The relationship between the theoretical backscatter TS and its measurement is given by:

$$TS = 20 \log \left| \frac{p_s}{p_i} \right|_{r=1, m},$$

$$= 20 \log \frac{\text{EV}}{\text{MBV}} - 20 \log \frac{R_{SH}}{R_{ST}R_{TH}} - (G_{\text{echo}} - G_{\text{mb}}), \tag{A.10}$$

where the terms MBV and EV refer to the respective "steady–state" main bang and echo voltage amplitudes measured from a particular hydrophone; the term  $20 \log \frac{R_{SH}}{R_{ST}R_{TH}}$  accounts for the total transmission loss in dB (each factor in this term is assumed to be divided by the reference distance of 1 m); and the difference in receiver gain settings for the echo versus main bang measurement is given by the G terms. Note that the TS measurement is independent of the source level and the hydrophone calibration coefficients as long as the main

bang and echo signals are processed from a the same receiver.

### 5.3 Test range layout

In Tables 5.2(a) and (b) the layout of the test range is specified for the Seneca-2 and Seneca-3 tests respectively. Four receivers (ITC #7001, 6" spherical ball), labeled H1–H4 were lowered over the edge of the TCP barge to the depths indicated in Tables 5.2(a) and (b). The hydrophones were first lowered over the side of the barge and the horizontal spacing on the surface measured relative to the PS (column H). The parametric source was deployed to a depth of 69 m (200 ft) using lengths of 6 in diameter steel pipe. The hydrophones were then lowered, resulting in the alignment indicated in Figure 5.1. In Column V of Tables 5.2, the vertical depth of each receiver (measured with a Durant Model 5–d–8–1–CL cable meter) relative to the PS source depth is indicated. Slight adjustments to the depth of each phone were made to insure that they were located on the main response axis of the PS.

The slant range distance  $R_{SH}$ , and thus the depth, of each hydrophone were accurately determined by performing several time-of-flight measurements of short pulses propagated from the PS using the 22 kHz carrier beam. The slant range distances between the source and each receiver  $(R_{SH} = \sqrt{H^2 + V^2})$  are summarized in column  $R_{SH}$  of Table 5.2.

In Table 5.2,  $\triangle t = t_{\rm echo} - t_{\rm mb}$  refers to the expected time delay for an echo return relative to the passing of the incident main bang pulse at a particular hydrophone. The transmission loss factors (TL) account for the 1/R de-

Table 5.2: Range layout and transmission loss factors.

#### (a) Seneca-2

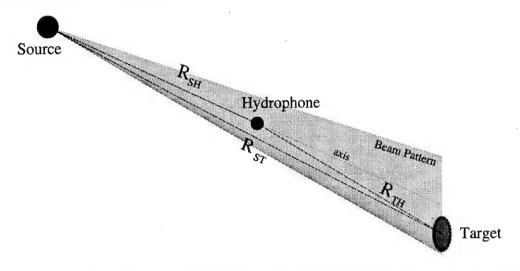
	Н	V	$R_{ m SH}$	$\triangle t$	$\theta$	$TL_{ m SH}$	$TL_{ m ST}$	$TL_{ m TH}$	TLF
	(m)	(m)	(m)	(ms)	(deg)	(dB)	(dB)	(dB)	(dB)
PS	00.00	00.00	00.00						
НЗ	17.90	13.02	22.13	33.26	36.03	26.90	33.21	27.47	-33.78
H2/CS2	23.53	17.01	29.03	23.55	35.86	29.26	33.21	24.47	-28.43
H1	28.92	20.73	35.58	14.34	35.63	31.02	33.21	20.16	-22.35
Target	37.20	26.67	45.77		35.64				
H4	39.21	26.67	47.42		34.22				
Bubbler	37.20	30.69	48.23		39.53				

#### (b) Seneca-3.

	Horiz	Vert	$R_{ m SH}$	$\triangle t$	$\theta$	$TL_{ m SH}$	$TL_{ m ST}$	$TL_{ m TH}$	$\operatorname{TLF}$
	(m)	(m)	(m)	(ms)	(deg)	(dB)	(dB)	(dB)	(dB)
PS	0.00	0.00	0.00						
Н3	17.83	12.78	21.94	33.19	35.62	26.82	33.17	27.46	-33.80
H2	23.34	16.73	28.72	23.65	35.63	29.16	33.17	24.51	-28.52
H1	28.72	20.58	35.33	14.35	35.63	30.96	33.17	20.17	-22.38
Target	37.01	26.52	45.53		35.62				
H4	43.07	30.86	52.98		35.63				
Bubbler	37.01	30.21	47.77		39.22				

Note: The items deployed are listed in the first column. Column–H lists the horizontal separation of each item relative to the PS location; Column–V lists the depth of each item beneath the PS; Column–R lists the direct path between the PS and each element;  $\Delta t$  is the expected time delay between the reception of incident and scattered pulse at a particular hydrophone;  $\theta$  is the declination angle between the PS and the element, relative to the horizontal. The transmission loss factors for the distances  $R_{SH}$ ,  $R_{ST}$ ,  $R_{TH}$  and the total transmission loss factor, TL are listed in the remaining columns (refer to Figure 5.4).

crease in signal amplitude due to spherical spreading of the signal between the source-hydrophone  $(R_{SH})$ , source-target  $(R_{ST})$ , and target-hydrophone distances  $(R_{TH})$ , as indicated in Figure 5.4. (The sonar equations which describe the transmission loss factors and the backscatter target strength measurement are presented in Appendix A.)



**Figure 5.4:** Scattering geometry for a single hydrophone and the target not in the backscatter position.  $R_{SH}$  is the distance between the source and the receiver;  $R_{ST}$  is the distance between the source and the target; and  $R_{HT}$  is the distance between the target and the hydrophone. When the target is in the backscatter position,  $R_{ST} = R_{HT} + R_{SH}$ .

The transmission loss factors between the source-target and target-hydrophone are listed Columns  $TL_{ST}$  and  $TL_{TH}$  of Table 5.2, where  $TL_{ST} = 20 \log_{10}(R_{ST})$  and  $TL_{HT} = 20 \log_{10}(R_{TH})$ . The total transmission loss factor for the target in the backscatter position is  $TLF = 20 \log_{10}(\frac{R_{SH}}{R_{ST}R_{TH}})$  are given in Column TLF. For the case of a rising bubble cloud, however, the target is not stationary. The bubble cloud is assumed to rise vertically (see Section 4.3.1), and thus the separation distances  $R_{ST}$  and  $R_{TH}$  (and therefore, the TLF) are

simple functions of the rise time and vertical rise velocity of the bubble cloud (horizontal drift was not accounted for).

The test range parameters given in Table 5.2 describe the special case when the bubble cloud crosses the axis of the test range (i.e., backscattering). However, since the bubble cloud position is not held fixed (since it is rising), the transmission loss factors that are needed to determine the time-evolving target strength are functions of cloud position, and therefore, functions of time relative to the cloud release. In general, the total transmission loss factor for the experimental geometry described here is given by  $TLF = TL_{SH} - TL_{ST} - TL_{TH}$ , where the subscripts SH, ST, and TH refer to the distance between source-receiver, source-target, and target-receiver, respectively (for example  $TL_{ST} = 20 \log(R_{ST})$ ). For a rising bubble cloud, however, the range distances  $R_{ST}$  and  $R_{TH}$  can be determined as a function of time relative to cloud release, and therefore, the total transmission loss factor is calculated at each increment of time. Assuming that the cloud rises vertically, then the distances  $R_{ST}$  and  $R_{TH}$  are given by

$$R_{ST} = \left[ (X_{\text{bubbler}})^2 + (Y_{\text{bubbler}} - u_{\text{rise}} \cdot t)^2 \right]^{\frac{1}{2}}$$
 (5.1)

$$R_{TH} = \left[ (X_{\text{bubbler}} - X_{\text{rec}})^2 + (Y_{\text{bubbler}} - u_{\text{rise}} \cdot t - Y_{\text{receiver}})^2 \right]^{\frac{1}{2}}, \quad (5.2)$$

where the X and Y coordinates are given in Table 5.2 relative the PS origin, and  $u_{\rm rise} = 0.39 \, {\rm m/s}$ . Shown in Figure 5.5 are the calculated rise-time dependent total transmission loss factors used to determine the target strength. Note that at  $t \approx 11 \, {\rm s}$ , the total transmission loss factor corresponds to the

values given in Tables 5.2(a) and 5.2(b), respectively.

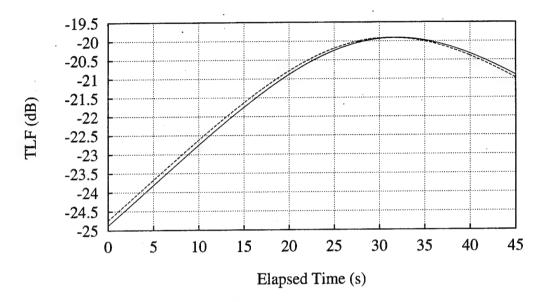


Figure 5.5: Total transmission loss factor for receiver H1 as a function of bubble cloud rise—time relative to cloud release. The dashed line corresponds to the total transmission loss correction applied to the Seneca-3 data, and the solid line for the Seneca-2 data. When the cloud is in the backscatter position,  $t \approx 11 \,\mathrm{s}$ , the TLF corresponds to the values given in Table 5.2.

## 5.4 Signal conditioning and recording

The receivers deployed axially along the PS beam were used to measure the incident main bang signal and scattered or *echo* signals. A schematic illustrating the instrumentation used to acquire the data from a single hydrophone is shown in Figure 5.6. Its operation is described as follows: First, the source was configured to send a steady stream of pulses of specified pulse width and difference frequency  $(f_{\triangle})$  at a repetition rate of either 1 or 2 Hz. The main bang signals were received on each of the hydrophones H1–H3, and continued

to propagate toward the target. The target responds by scattering sound in all directions, and the backscattered echo contribution (EV) was recorded from hydrophones H1–H3. In order to remove the large amplitude PS primaries ( $\approx 22\,\mathrm{kHz}$ ) from the received waveforms, a 2 kHz passive low–pass filter (a single pole RC filter with -6 dB/octave insertion loss) was placed inline prior to pre–amplification; during the conventional source tests these filters were removed. The signals acquired at each hydrophone were fed into an Ithaco (model 453) pre–amplifier set to variable gain and including a 100 Hz high pass filter. The pre–amplifier gain was adjusted to maximize the main bang and the echo voltages generated by each hydrophone within a  $1\,\mathrm{V}_{\mathrm{peak}}$  range (the input range of the recording tape deck).

Following pre–amplification, a KronHite (model #3343) active low pass filter set one to two decades above the difference frequency was installed to insure that the signal recorded on the tape deck lacked frequency components outside the range of the recorder at its lowest possible recording speed at a given driving frequency. In addition, this filter served as an anti–aliasing filter, since the tape recorded data was to be digitized after returning to the laboratory. The incoming main bang and echo signals from hydrophone H1 were monitored in situ using a LeCroy 9450 digital oscilloscope in order to confirm correct range alignment and to confirm that scattering did indeed occur as the cloud ascended through the main lobe of PS beam.

The signals were recorded on a 3/4-inch reel-to-reel tape deck (Teac SR31–DF) with 7 recording tracks. The tape deck was capable of operating at one of nine variable speeds and each channel was recordable in either a FM or Direct

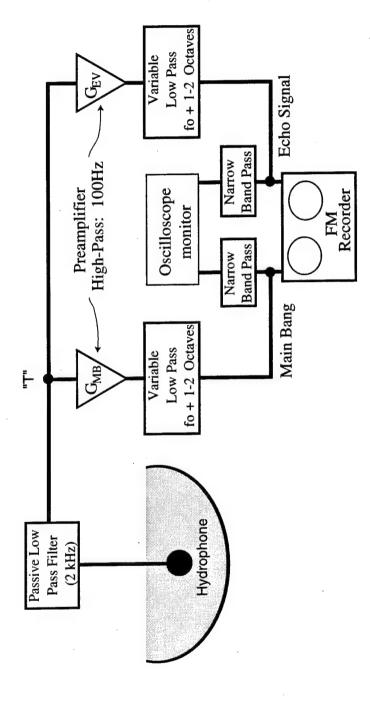


Figure 5.6: Instrumentation setup for a single hydrophone during the Seneca-3 experiment. The large amplitude  $\approx 22\,\mathrm{kHz}$  primaries from the parametric source were removed using the  $2\,\mathrm{kHz}$  passive low-pass filter. Then the signal was split into two channels using a "T" and fed into a pair of pre-amplifiers and active low-pass filters gain was set to optimally amplify either the incident or scattered echo pulse respectively in a  $1 V_{peak}$  range prior to recording. During the Seneca-2 experiment, the main bang and echo signals were recorded from separate data runs, thus the "T" was not inserted. Online processing was performed by monitoring narrow band passed signals prior to recording to individual main bang and echo channels on the reel-to-reel tape deck. The pre-amplifier on a LeCroy 9450 digital oscilloscope.

recording mode. The frequency response and signal to noise ratio for these operating speeds is shown in Table 5.3. For the majority of the data recorded, the tape speed was set to 38 cm/s however, tapes speeds of 19 and 76 cm/s were also utilized. For the echo measurements, the preamplifier gain was set high enough such that the incident main bang signal saturated the amplifier output and the tape deck input. This did not have a detrimental effect on the recorded data as the equipment had sufficient time to recover from these saturations.

	]	FM mo			Γ	irect n	node				
Speed	Freq	uency ]	Range	S/N ratio	S	peed	Frequency Range				
(cm/s)	'	(kHz)	)	(dBrms)	(c	m/s)	(Hz)		(kH		
152.40	DC		40	49	152	2.40	300		300		
76.20	DC		20	49	76	5.20	200	-	150		
38.10	DC		10	49	38	3.10	100		75		
19.05	DC		5	48	19	0.05	100		37		
9.52	DC		2.5	46	9	).52	100		18		
4.76	DC		1.25	44	4	1.76	100		9		
2.38	DC		0.625	44	2	2.38	100		4.5		
1.19	DC		0.313	42	ī	.19	100		2.25		
0.59	DC		0.156	40	0	).59		-			

Speed	Freq	S/N ratio		
(cm/s)	(Hz)		(kHz)	(dBrms)
152.40	300		300	30
76.20	200		150	30
38.10	100		75	30
19.05	100		37	30
9.52	100		18	30
4.76	100		9	30
2.38	100		4.5	30
1.19	100		2.25	30
0.59		_		_

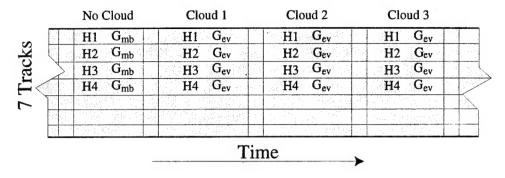
Table 5.3: Tape deck (Teac SR31-DF) specifications.

Shown in Figure 5.7(a), is a cartoon of a tape segment illustrating a sequence scattering measurements recorded to tape for a particular driving frequency. The reel-to-reel recorder were capable of recording 7-tracks of data simultaneously, only the first four are shown in the figure. During the Seneca-2 experiment, a recording of the incident main bang signal was made from all hydrophones prior to a series of cloud releases. The pulse repetition frequency was set to 2 Hz, the preamplifier gain was set to  $G_{\rm mb}$  (arbitrary, but dependent on the driving frequency) and roughly 60 pulses or 30 s of data were recorded to the analog reel-to-reel tape. At this gain setting, the specular echo from the target often appeared to be buried in the background noise.

Prior to recording the sequence of scattering measurements from rising bubble clouds, the preamplifier gain  $(G_{\text{echo}})$  was adjusted in order to maximize the specular return of the backscatter signal within a  $1\,\mathrm{V_{peak}}$  range. Typically, the difference in preamplifier gain settings  $(\Delta G = G_{\text{echo}} - G_{\text{mb}})$  exceeded 20 dB, or a factor of 10 in signal amplitude. After determining the appropriate gain setting for the backscatter measurement at a given driving frequency, a minimum of three cloud releases were recorded to the reel-to-reel tape units. Following this sequence of measurements, the source frequency was modified and the process repeated. In Appendix B, a table listing the recording parameters and test schedule for the Seneca-2 test can be found.

In contrast, during the Seneca–3 test the main bang and backscatter echo signals were recorded simultaneously as illustrated in Figure 5.7(b). This obviously required double the number of pre–amplifiers and filters, but less overall time per frequency run (recall Figure 5.6). In addition, the advantage of this recording configuration is a 1:1 correspondence between the main bang pulse and the backscattered echo pulse. In Appendix C, a table listing the recording parameters and test schedule for the Seneca–3 test can be found. The data tapes from both experiments were digitized and processed after returning to the lab. In Chapter 5, the digitizing process and subsequent data analysis is described.

In Figure 5.8, a photograph of the main equipment rack used during this experiment is shown. It contained the pre–amplifiers (bottom) and a set of three active low–pass filters (KronHite #3343) used to condition the incom-



(a) Seneca-2 tape segment.

	Cloud 1				Cloud 2				Cloud 3				Clo			
7.0		H1	$G_{mb}$		H1	G <sub>mb</sub>			H1	$G_{mb}$	A.47	A 5.14	H1	G <sub>mb</sub>		100
racks		H1	$G_{ev}$		H1	$G_{ev}$			HI	$G_{ev}$			H1	Gev	1, 1	
ac	7	H2	$G_{mb}$		H2	$G_{mb}$			H2	$G_{mb}$	, Argi		H2	Gmb		
_≒<		H2	Gev		H2	Gev			H2	$G_{ev}$			H2	Gev		
_			- 1147										134			
													4,415			
										31.						
	Time															

(b) Seneca-3 tape segment.

Figure 5.7: Sample reel-to-reel tape segments (elapsed time) for the Lake Seneca experiments. The tape deck was capable of recording 7 data tracks simultaneously and the signals recorded to each track is listed above. The signals recorded and preamplifier gain settings used are indicated here only for tracks 1-4. Signals were recorded to tape for approximately 60 s per cloud release. In the region between data runs a 2 s calibration tone was recorded. Note: Due to the instrumentation setup used in the Seneca-2 effort, a single main bang recording was conducted prior to a series of cloud releases. During the Seneca-3 experiment, the main bang and echo recordings were made simultaneously.

ing signals prior to recording on the analog tape deck. Above the recording filters are located a trio of WaveTek #753A "Brick-Wall" filters which were configured to narrow band-pass the conditioned signals in order to monitor

the backscattering results "on–line" with the LeCroy 9450 digital oscilloscope. Above the oscilloscope is a signal generator and the *Deep-Thought* bubble maker controller unit. On top of the equipment rack is a digital volt meter used to monitor the differential pressure transducer mounted inside the bubble maker.

# 5.5 Test range calibration

Prior to lowering the bubble maker and performing backscattering measurements from rising bubble clouds, a test target was deployed in the expected backscatter position of the test range in order to verify range alignment (recall Figures 5.1 and 5.4). This calibration was important in order to correctly account for the transmission loss factors needed to determine the target strength in Eq. (A.10), and to ground truth the experimental procedure. In order to determine the total transmission loss factor (TLF), the distance between the source-hydrophone  $(R_{SH})$ , source-target  $(R_{ST})$ , and target-hydrophone  $(R_{TH})$  need to be known accurately for each receiver.

The test target was a hollow (air-filled) steel sphere of radius  $a=1.12\,\mathrm{m}$  weighted with 800 lbs of ballast (lead and steel) and lowered to the backscatter position at a depth of 87.6 m (depth determined using the cable meter). To determine the distances  $R_{ST}$  and  $R_{TH}$ , the time delay between the reception of an incident main bang pulse and scattered return (specular reflection) from the steel sphere was measured and compared to the expected values listed in Column  $\Delta t$  of Table 5.2. For Hydrophone 1 (H1), the expected time delay

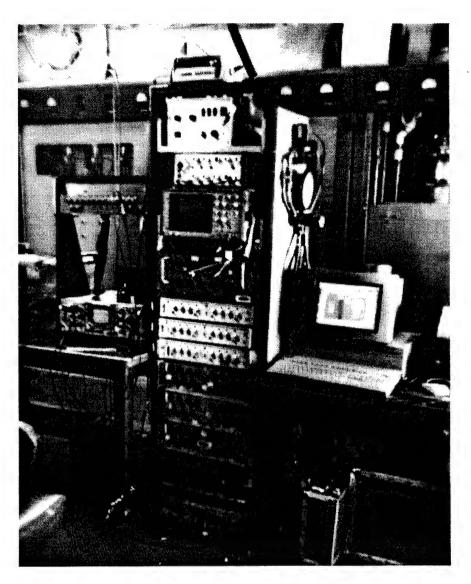


Figure 5.8: Photograph of instrumentation setup. Located on the top of the equipment rack is a digital voltmeter used to monitor the internal over pressure of the bubble maker. Beneath this, the *Deep Thought* bubbler controller unit is mounted in the upper most location of the equipment rack. A signal generator and the LeCroy 9450 digital oscilloscope are next, followed by a bank of three WaveTek #753A "brick-wall" filters, and three KronHite #3343 filters set to low-pass mode. Finally, the Ithaco pre-amplifiers are located at the bottom of the rack. The computer on the right of the equipment rack was used to remotely control the recording tape deck and digitize and store the outputs from the precision electronic mass flow meter and the pressure transducers mounted in the bubble maker.

was  $\triangle t_{\rm expected} = 14.3\,\mathrm{ms}$  and the measured time delay  $\triangle t_{\rm meas}$  varied roughly between 13 and 16 ms. The variation is presumably due to slight currents in the lake which cause either the hydrophones or the target to move slightly off axis. Because of the agreement between the measured and expected time delays, it was concluded that the test range was laid out to the best of our ability. The locations and expected time-delays for the sources and receivers hydrophones deployed during this experiment are indicated in Table 5.2.

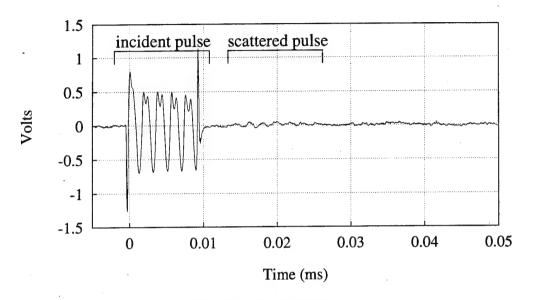
Aside from calibrating the test range, the steel target allowed us to test our ability to perform a quantitative backscatter TS measurement using Eq. (A.10). The PS was configured to send 12 ms pulse of sound every 1 s at difference frequency  $f_{\triangle} = 500\,\mathrm{Hz}$  and pulse length of 10 ms. The incident main bang and backscattered echo signals were recorded to analog reel-to-reel tape for each receiver in the backscatter direction using the method described in Section 5.4. An example of a single incident main bang and scattered echo pulse recorded from receiver H1 (the nearest hydrophone in the backscatter direction) is shown in Figure 5.9. The echo arrived at the receiver  $\Delta t = 14.5 \, \mathrm{ms}$ after the reception of the leading edge of the incident pulse. The preamplifier gain setting for the incident main bang channel was  $G_{\rm mb}=46\,{\rm dB};$  and for the echo return, the quantity of  $G_{\text{echo}} = 66 \,\mathrm{dB}$  gain was added. The rms voltage amplitude was measured over an 8 ms "steady-state" portion of the received main bang and scattered echo pulse wave forms (e.g., MBV =  $0.27\,\mathrm{V_{rms}}$  and  $\mathrm{EV} = 0.09\,\mathrm{V_{rms}})$  after the data streams were band–pass filtered at  $500 \pm 50\,\mathrm{Hz}$ (Note: Here the term "steady state" refers to the middle portion of the pulse). These voltages were entered into the equation for the target strength (Eq.A.10)

along with a transmission loss factor of  $-22.38\,\mathrm{dB}$  and the pre-amplifier gain factors in order to determine TS. In this example, for a driving frequency of  $500\,\mathrm{Hz}$ , the measured TS by receiver H1 is  $-7.4\,\mathrm{dB}$ .

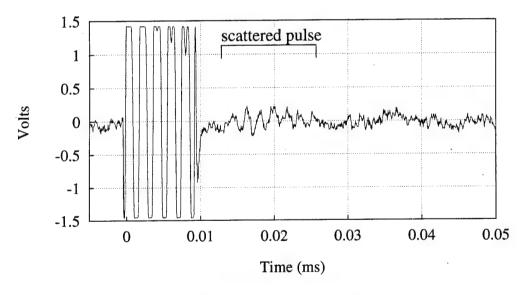
For the PS driven at  $f_{\triangle} = 5000\,\mathrm{Hz}$ , the measured TS was  $-12.3\,\mathrm{dB}$ . This is in close agreement with the expected TS of a perfectly reflecting sphere of the same radius at high frequencies, -11.1 dB (recall that  $\mathrm{TS}_{\mathrm{rigid}} = 20\log_{10}(a/2)$ ). The comparison of the theoretical TS of a perfect reflector at high frequencies to the hollow steel spherical shell is likely valid (Urick, 1967) and suggests that our ability to measure the TS is good.

## 5.6 "On-line" data processing

In this section, the preliminary Lake Seneca "on–line" analysis of the target strength measurements from rising bubble clouds is presented. As shown in Figure 5.6, a LeCroy 9450 digital oscilloscope was utilized to monitor the received and pre–processed signals. The primary reason to conduct on–line monitoring is to insure that the experimental measurements and associated equipment are performing adequately. Another motivation for conducting this analysis was to obtain a rough estimate of the frequency dependent target strength and to establish the viability of the parametric source as a research tool which could be utilized to conduct scattering experiments from bubble clouds. Thus comparisons between the target strength measurements utilizing both the parametric source and the conventional sources were made over a range of frequencies spanning 500 Hz to 7 kHz. Below 500 Hz, the conventional



(a) Main Bang waveform



(b) Backscattered waveform

Figure 5.9: Incident main bang (a) and backscattered (b) waveforms from the steel target sphere measured at receiver H1. Here, the driving frequency is 500 Hz and the backscatter return is received at  $\Delta t_{\rm meas} = 14.5$  ms, relative to the incident pulse. To enhance the echo, an additional 20 dB of gain was added to (b).

tional sources were ineffective due to low source level, and thus the signal to noise ratio and reverberation increased correspondingly. The parametric source, however, was utilized down to 250 Hz, at which point it too suffered from a significant drop in signal level. For the frequency range between 500 and 3000 Hz, the scattering results using both sources were found to be in agreement.

Shown in Figure 5.6 is an illustration of the signal processing instrumentation used to prepare raw signals received at each hydrophone. While conducting the experiment, receiver H1 was monitored and processed on—line in the following manner:

Once the range was aligned and calibrated and the source frequency and pulse duration set, the bubble–maker solenoids were activated for 3.4 s, resulting in a rising bubble cloud. The cloud rose from the bubbler and ascended through the source beam axis yielding a time–varying backscattered echo level. Because the rise velocity of the bubble cloud was not measured until the final day of the experiment, it was assumed that the bubble cloud crossed the axis of the test range when a peak in the scattered voltage amplitude occurred. This location of the peak varied by as much as 6 s depending on the driving frequency. For the lower frequencies (250 to 600 Hz) the peak occurred at approximately 6 to 8 s after cloud release, and for higher frequencies (> 1000 Hz) this peak occurred at about 12 s after cloud release. The latter is in good agreement with the expected time needed for a cloud to cross the backscatter location after cloud generation, while the former is presumably due to several factors, most notably the decreasing signal to noise ratio as the source driving

frequency was lowered. This decrease is due to the fact that the signal level introduced into the test range by the parametric source (or nearly any source, for that matter) decreased as the frequency was lowered.

Furthermore, at some frequencies, there were multiple peaks. This would seem to suggest that: (a) angular dependent scattering had occurred, (b) perhaps interference with the bubble maker was a problem, or (c) the bubble clouds generated were not rising in a repeatable manner. The latter conclusion was ruled out after performing the rise time measurements. It was clear then that our initial assumption (that the cloud crossed the axis of the test range when the peak scattering level was measured) was in error. It should be noted, however, that for a given driving frequency the elapsed-rise-timedependent scattering amplitudes for several subsequent cloud releases were well correlated. The pulse length, and thus pulse bandwidth, rather than the number of cycles was held fixed for the various driving frequencies during the experimental effort. The reason for this is that at the lower driving frequencies, we wanted to insonate the target with as much energy as possible. However, due to the spacing between the first hydrophone (H1) and the bubble cloud, the maximum pulse length was limited to 14 ms (for longer pulses, an overlap in the main bang and scaterred signals would occurr).

For the majority of the scattering data runs the incident pulse duration was set to 12 ms. The pulse repetition rate was set to 2 Hz for the Seneca-2 test and 1 Hz during the Seneca-3 test. The oscilloscope was set to trigger on the leading edge of the incident main bang pulse. Range gating of the echo signal was performed by configuring the oscilloscope to capture and store

a 10 ms window centered on the expected location of the backscatter pulse waveform. For receiver H1, the 10 ms echo—gate was fixed to acquire data from 15.5 to 25.5 ms after the leading edge of the incident main bang waveform. A sequence of the range—gated echo voltage returns, one for each incident pulse propagated across the test range during the ascent of a single bubble cloud, were stored in the oscilloscope memory. After the cloud had risen for approximately 45 s, sufficient time for it to exit the test range, the sequence of stored backscatter waveforms were recalled from memory, and displayed on the oscilloscope monitor.

Finally, for at least one bubble cloud release at each driving frequency, a hard copy of the LeCroy oscilloscope display screen was made. In Figure 5.10, an image of the LeCroy console containing the time-elapsed, time-gated backscatter echo from resulting from a single rising bubble cloud is shown for a driving frequency of 3.0 kHz. As expected, the echo voltage amplitude varied as the cloud traversed through the source beam, reaching a maximum 10 s after release. As the bubble cloud continued to rise, the backscatter echo voltage amplitude eventually diminished after exiting the test range. Individual scattering returns (bins) in the echo time sequence in Figure 5.10 could be displayed and were used to determine the target strength of the bubble cloud.

A similar sequence of time–gated main bang waveforms were also stored. The maximum rms voltage amplitude of the echo in a time sequence (EV) and the average rms amplitude of the main bang sequence (MBV) were recorded in the experimental log book along with the preamplifier gain settings. Finally, using Eq. (A.10), the target strength was calculated (also recall Table 5.2).

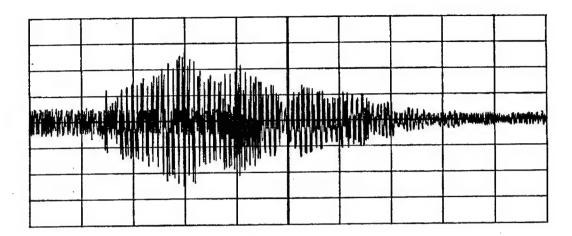


Figure 5.10: Time gated echo response from a rising bubble cloud measured by receiver H1. Here, the PS was driven at  $3000\,\mathrm{Hz}$ . The time duration of this oscilloscope trace is  $50\,\mathrm{s}$ . For each incident main bang pulse, a  $10\,\mathrm{ms}$  window centered over the expected backscatter signal position (between  $15.5\,\mathrm{and}\,25.5\,\mathrm{ms}$ ) after the leading edge of an incident main bang pulse) was digitized and stored to memory. The pulse repetition frequency was  $2\,\mathrm{Hz}$  and thus  $100\,\mathrm{such}\,$  echoes are displayed in a compressed form ( $10\,$  echoes per division). The cloud was released within  $\pm 1\,\mathrm{s}$  of the initial trace and the peak echo response occurs between the  $10-15\,\mathrm{s}\,$  mark.

In Figure 5.11, the preliminary target strength measurements for frequencies below 1600 Hz are shown. These data sets were conducted utilizing the conventional source (CS1: 500 to 1250 Hz), and the parametric source (250 to 1600 Hz). The conventional source data was collected during the Seneca-2 test and the target strength increased as the driving frequency was lowered from 700 Hz down to 350 Hz. This increase in the target strength was indicative of a resonance at a lower frequency and these promising results were the prime motivation for reconditioning the parametric source and performing the Seneca-3 tests (recall that during the Seneca-2 experiment the parametric source was usable only down to 800 Hz). Unfortunately, the conventional sources could not be driven below 350 Hz due to decreasing signal level. The

substantial error—bars and lack of agreement with the PS measurements in the conventional source measurements are due to excess reverberation; clearly at the higher frequencies this was less of a problem.

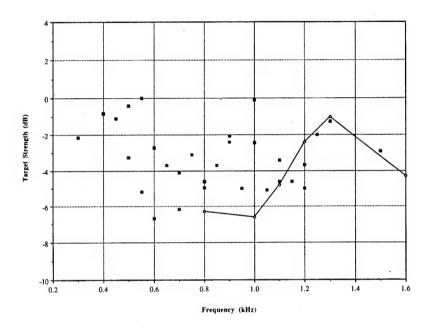


Figure 5.11: Backscatter target strength measured below 1600 Hz. The hollow squares were obtained using the parametric source; the solid squares using the conventional source. The error bars indicate the extent of deviation in the backscatter target strength over a consecutive series of three bubble cloud releases. Note: The curve joining the points is only used to train the eye.

The peak centered at 300 Hz and having TS  $\approx -1\,\mathrm{dB}$  is strongly suggestive of a resonance. Recall that the measured TS of the 1.12 m-diameter steel sphere at 500 Hz was only -7 dB. Even though it was not known until a later date that the dimensions of the bubble cloud were on the same order of magnitude of the steel sphere, it still seems remarkable that the measured TS was so large. From Figure 3.2, a rigid sphere of this dimension (ka=1.1) would possess a target strength of  $\approx -13\,\mathrm{dB}$ . Incidentally, the measured tar-

get strength of the hollow steel sphere for a driving frequency of 500 Hz is approximately -7 dB—the difference in the measured and predicted value for the steel sphere is hypothesized to be due to the fact that at 500 Hz, this sphere does not appear to be impenetrable (at large ka it does), or it is even more likely that at this low frequency in which the source level decreases dramatically, that the reduced signal—to—noise ratio made an accurate measurement impossible.

For the bubble cloud characterization described in Section 4.3 and for frequencies below 600 Hz, the Lake Seneca bubble cloud is an acoustically compact scatterer, and thus can be treated as a sphere of radius  $a=0.376\,\mathrm{m}$  with void fraction  $\beta=0.0055$ . Such a cloud would possess a resonance frequency of  $\Omega_0\approx 326\,\mathrm{Hz}$  and monopole  $TS\approx -3\,\mathrm{dB}$  at a depth of 87.6 m, according to Eq. (3.32) and (3.30). Although the preliminary measurement analysis did not agree quantitively with the model predictions, the presence of the scattering peak at approximately the predicted frequency (300 Hz) in the Figure 5.11 is encouraging.

# 5.7 Post-experimental analysis

### 5.7.1 Digitizing the data tapes

The data tapes were digitized and processed as illustrated in Figure 5.12. The tapes were mounted on the tape deck for playback. Recall that prior to recording the signals, the waveforms were pre-conditioned (low-pass filtered one to two decades above the driving frequency and 100 Hz high-pass filtered). How-

ever, to insure that no signal aliasing occurred during the digitizing process, the tape deck output was filtered at one-half the data acquisition rate (dependent on the driving frequency). The filter used was a Stewart Brickwall Model VBF-8 with 132 dB rolloff per decade. The amplitude of each output channel on the tape deck was calibrated by adjusting the output level of a pre-recorded calibration tone to  $1\,\mathrm{V_{peak}}$  as suggested in the Teac SR31-DF manual. This two-second calibration tone was automatically recorded prior to each data run during the experiment. Also, a sequence of calibration tones were recorded to the tape deck each morning during the experimental effort.

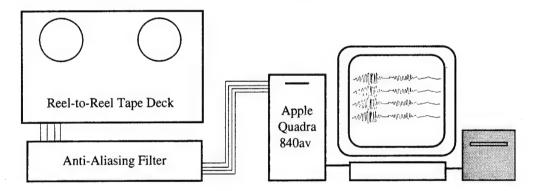


Figure 5.12: The data tapes were mounted on the reel-to-reel tape deck and digitized. Prior to digitizing, an anti-aliasing filter set to one-half the data acquisition rate was placed between the tape deck output and the digitizer input. The computer (Apple Macintosh Quadra 840av) was outfitted with a National Instruments NB-MIO-16 multifunction I/O board and performed the waveform digitizing. The digitized waveforms were stored to files on magneto-optical disks for future processing.

For the majority of the recorded data, the National Instruments NB–MIO–16 expansion board (maximum digitizing rate, 100 kHz) in the Apple Macintosh Quadra 840av computer was configured to digitize data at a rate of at least 10 points per cycle, however, the sampling rate often exceeded 20

points per cycle. For frequencies below 1000 Hz, the sampling rate was set to 20 kHz, and four channels of the tape deck output were digitized simultaneously.

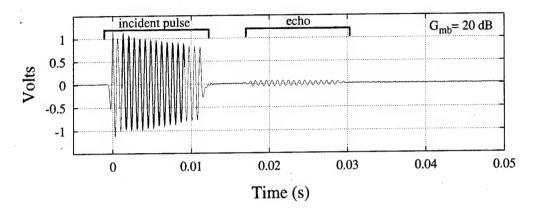
Shown in Figure 5.13 is a pair of digitized main bang and echo wave forms recorded from receiver H1 for a source driving frequency of 1400 Hz during the Seneca–3 experiment. The waveform in Figure 5.13(a)was preamplified at  $G_{\rm mb}=20\,{\rm dB}$ , and in Figure 5.13(b) the gain was set to  $G_{\rm echo}=43\,{\rm dB}$ , in order to maximize the respective main bang and scattered echo signal in a  $\pm 1\,{\rm V}_{\rm peak}$  range (the input range of the recording tape deck).

The software used to control the digitizing process, National Instruments LabView 3.0, was configured to capture 150 ms of data after pretriggering 10 ms prior to an incident main bang pulse. Recall that for receiver H1, the backscattered echo was expected to arrive approximately  $14.3\pm2.0$  ms after the leading edge of the main bang pulse. For frequencies below 1 kHz, the main bang from receiver H1 was used as the trigger channel. Once the voltage output of this channel exceeded the specified amplitude, all four channels were simultaneously digitized. For the higher frequencies, different output channels were used to trigger the digitizing process. The computer software controlling the digitizing process was initiated manually (*i.e.*, a simple "mouse click") at the point of cloud release by monitoring the playback of a single tape deck output channel on an external speaker (the activation of the solenoid bank resulting in the release of a cloud from the bubble maker was audible).

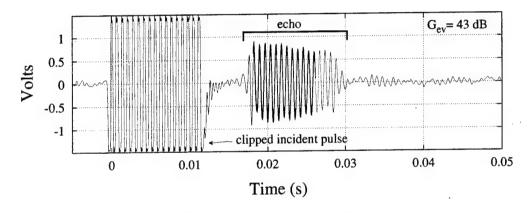
The digitizing process continued until at least  $30\,\mathrm{s}$  had passed and was stopped prior to the  $60\,\mathrm{s}$  mark, or when the next cloud was released from

the bubble maker. The bubble cloud rise velocity was 0.39 m/s, thus in 30 s a bubble cloud should rise approximately 11 m above the bubble maker. For the Seneca-2 data tapes, at least 60 waveforms per data run were digitized (i.e., the pulse repetition rate was 2 Hz); and for the Seneca-3 data tapes, at least 30 waveforms were acquired for each data run. The digitized signals were stored to files on magneto-optical disks for future analysis. This process was repeated for each driving frequency, and each cloud release recorded to the reel-to-reel tapes.

Shown in Figure 5.14 is the echo voltage response measured by receiver H1 for a single rising bubble cloud at a source driving frequency of 1400 Hz during the Seneca–3 experiment. Consecutive waveforms (bins) are separated by 1s, and are to be read down first, then across. The length of each time trace in Figure 5.14 is approximately 50 ms. The bubble cloud was generated within  $\pm 0.5 \, \mathrm{s}$  of the upper left waveform. The scattering response increased as the bubble cloud traversed through the axis of the source beam, reaching a maximum in bins 11–14. This time–frame is within the 3.8 m (backscatter position) rise–time measurements discussed in Section 4.3.1. The clipped pulse early in each bin is the incident main bang waveform. The reason for the clipping is due to the enhanced pre–amplifier gain applied to the received signal in order to maximize the echo pulse in a  $\pm 1 \, \mathrm{V}_{\mathrm{peak}}$  range prior to recording. A separate channel with the pre–amplifier gain set to maximize the main bang pulse in this voltage range was also recorded. Typically, the difference in gain settings between the main bang and echo channels exceeded 20 dB.



(a) Incident main bang signal



(b) Backscattered echo signal

Figure 5.13: Sample incident and scattered pulse ≈ 11 s after cloud release (backscatter position) for a driving frequency of 1400 Hz and measured at H1. To increase the resolution of the scattered echo, the signal was pre–amplified 43 dB prior to recording. The main bang signal was pre–amplified separately by 20 dB. The incident pulse length is 12 ms. The darker region in each plot corresponds to an 8 ms window over which the rms measurements were made. In this case, the rms voltages are MBV =  $0.6 \, V_{\rm rms}$ , EV =  $0.46 \, V_{\rm rms}$ . Thus for a total transmission loss factor of  $TLF = -22.38 \, \rm dB$  (recall Table 5.2) and the preamplifier gain settings listed above, the measured target strength as given by Eq. (A.10) is TS =  $-2.9 \, \rm dB$ .

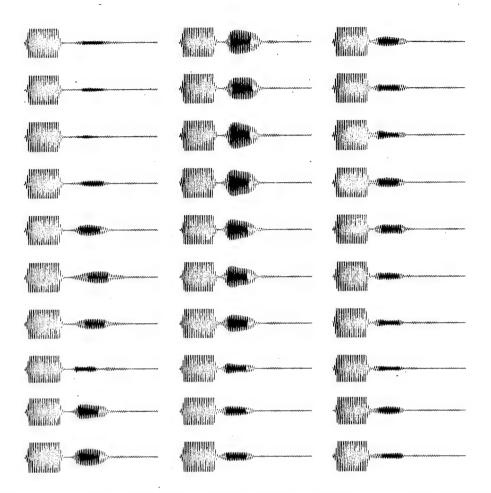


Figure 5.14: Echo response (voltage versus time) after a cloud release at a driving frequency of 1400 Hz. Consecutive time traces are separated by 1 s and are to be read down first then across. Each trace is 50 ms in duration. The cloud is released within  $\pm 0.5$  s the upper left signal. As the cloud rises through the target area, significant scattering is observed with the expected backscatter position in the upper four traces in the middle column. The darker region (8 ms) contains the maximum rms voltage in the "steady–state" portion of the echo and is used for the TS measurements. The larger amplitude 'clipped' incident main bang ping is due to optimal amplification of the echo.

### 5.7.2 Data analysis

Because of the relatively low Q (wide bandwidth) of the incident pulses at the lower frequencies, the archived data files were processed using two methods.

The "steady-state" rms amplitude of the main bang and scattered waveforms were used to determine the target strength at each driving frequency using an algorithm similar to the on-line analysis. This method, however, only results in a single backscatter data point for each driving frequency, which is effectively the "average" value over the pulse bandwidth. For example, consider the target strength versus frequency curves shown in Figure 3.5, in particular consider the dashed curve with the sharp resonance peak at approximately 600 Hz. To resolve this high–Q resonance peak using amplitude measurements from broad-band (low–Q) pulse is impossible. Rather, this method would result in an "average" response, which tends to lower resonance peaks and increase the dips.

However, utilizing Fourier analysis techniques, the broad-band pulses can be used to determine the frequency response of the scatterer. Furthermore, when the bandwidth of the incident pulses at several different driving frequencies overlap (as is the case in this experiment), a much higher resolution frequency response curve can be obtained.

### Steady-state rms analysis

In performing the preliminary experimental analysis, the rise velocity of the bubble cloud was unknown and the peak rms amplitude of the main bang and scattered echo voltage provided by hydrophone H1 was used to determine the backscatter target strength of the bubble cloud. During the experiment, it was expected that the strongest scattering returns from the bubble cloud would arrive when the bubble cloud crossed the axis of the test range. The

expectation was that the scattering level would be lower prior to and after the cloud passed through the backscatter location. This was not observed for all driving frequencies. Rather, for some source driving frequencies, the amplitude of the scattered return was observed to increase many seconds prior to the time needed for the cloud to cross the axis of the source beam (12 s), decrease for a few seconds, and then increase again. These unusual observations were repeatable for each of the several clouds released during a given frequency run. Thus, it was clear that some unexpected phenomenon was effecting the scattering observations. At the time, it was postulated that either angular dependent scattering (i.e., not normal incidence backscatter) or perhaps acoustic interference with the bubble maker had occurred.

Another limitation in the preliminary analysis was that the transmission loss factor used to determine the TS level was fixed. This was due to the fact that only a single measurement, the backscatter target strength, was being performed. In this analysis, the rise-time dependent transmission loss factors from Figure 5.5 will be used to determine the elapsed time target strength.

Recall Figure 5.14 in which the evolution of the backscattered echo response of a single rising bubble cloud is displayed. The echo location in each waveform varies as a function of elapsed rise-time, and thus constant range-gating of the scattered echo cannot be used as it was during the "online" analysis. Moreover, even though consecutive scattering runs at a given driving frequency were highly repeatable, deviations in the echo delay of 1 to 2 ms were not unusual between cloud releases. These deviations are thought to be due to lake currents which result in the horizontal displacement of the

bubble cloud and/or the receivers.

The peak rms steady-state amplitude of the backscattered echo signal was measured for each incident pulse (see the darker region) in order to determine the target strength as a function of rise time. A simple algorithm was designed to locate the echo pulse in these waveforms. First, the rms amplitude over an 8 ms window beginning 35 ms after the trigger on the main bang pulse  $(t=0\,\mathrm{s})$  was calculated. Second, this measurement was compared with the rms amplitude over a window of the same width beginning 0.5 ms prior to the  $35\,\mathrm{m/s}$  delay. The second step of this procedure was repeated until the echo measurement window reached the tail of the incident main bang pulse, and then the peak rms echo amplitude was determined. This peak echo voltage amplitude, along with the average rms amplitude of the incident main bang pulse waveforms, the pre-amplifier gain factors, and the total transmission loss factors from Figure 5.5 were used to determine the target strength as a function of rise time using Eq. (A.10).

Shown in Figure 5.15 are the bubble cloud rise—time dependent target strength measurements for a consecutive sequence of three bubble cloud releases. For this data run, the source driving frequency was set to 1400 Hz, and the measurements were performed using receiver H1, the closest to the target. The average target strength for these three runs is indicated by the dashed line. It is clear that there is a significant deviation in the measured TS in the first few seconds after cloud release, and again after 20 s. This variability is likely due to the fact that the bubble cloud was not within the -3 dB beam—width of the parametric source. The variation is thereby casued by the

variability associated with the low signal to noise ratio conditions. However, in bins 5–15, the measured TS for the three clouds is highly repeatable since the bubble cloud is definitely crossing through the main lobe of the parametric source, resulting in an improved signal to noise ratio. In Appendix E, the remainder of the rise–time dependent target strength measurement sequences are illustrated.

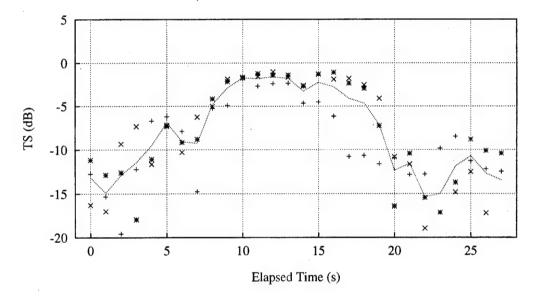


Figure 5.15: Time elapsed target strength for a driving frequency of 1400 Hz during the Seneca–3 experiment. The crosses, stars, and plus signs are the elapsed time dependent TS measurements for individual cloud releases. The boxed points and the dashed line joining these points indicate the average elapsed time TS. The minimum measurable signal in the absence of a rising bubble cloud was -26 dB.

In Figure 5.16 the elapsed time target strength averaged over three consecutive cloud releases is shown for each of the source frequencies used during the Seneca-3 experiment. This figure illustrates some important points. In the frequency range 250 to 500 Hz, the target strength of the rising bubble cloud is relatively strong during the first 20 s after cloud release. This is in part due

#### Composite TS SL3 H1

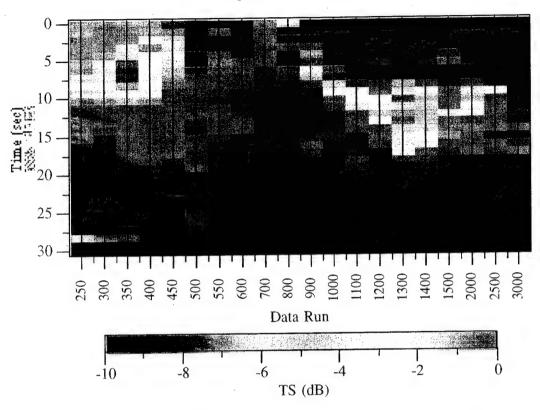


Figure 5.16: Summary rise—time dependent TS measurements. Here, the target strength is computed from the average echo voltage return from a sequence of three rising bubble clouds. The individual data runs are plotted in Appendix E.

to the lower signal to noise level at these frequencies compared to the higher frequency data sets. The PS is less efficient at the lower difference frequencies, and thus our ability to measure the scattered echo over the background spectrum was reduced. It is clear, however, that a peak in the TS versus frequency curve exists between 300 and 350 Hz at approximately 7s after cloud release, at least 3s earlier than expected.

The reason for the earlier than anticipated echo return could be due to

the long sonic wavelengths and the proximity of the bubble cloud to the bubble maker (interference phenomenon). It is plausible that the two are acoustically coupled, resulting in an enhanced echo return. The higher frequency data (> 1000 Hz), on the other hand, clearly demonstrates that the bubble cloud crossed the axis of the test range between 9 and 17s after cloud release. This is in good agreement with the measured rise-time data which suggest that the head of the cloud ascended to the backscatter target location in approximately 9s and the lower boundary of the cloud passed through this position nearly 14s after being released into the lake.

According to the rise-time measurements, the bubble cloud should cross the axis of the test range between 9 and 14 seconds after generation. Thus, to determine the backscatter TS, the rms voltage in bins 10–13 were smoothed and averaged over each of the three cloud releases, resulting in the target strength versus frequency curve given in Figure 5.17. The uncertainty in the backscatter TS measurement is derived from the maximum and minimum deviation with respect to the averaged value and are not determined statistically (due to the limited number of observations).

Figure 5.17 plots the smoothed-averaged backscatter target strength measurements at each source driving frequency below 1600 Hz along with the prediction from the simple model presented in Chapter 3. The solid line is a calculation of the target strength using the asymptotic expression (Eq. 3.30), and the dashed line a calculation using the full wave solution of Eq. (3.14). The non-acoustically determined parameters,  $P_o = 9.82 \times 10^5 \,\mathrm{Pa}$ ,  $c = 1421.5 \,\mathrm{m/s}$ ,  $\beta = 0.0055$ , and  $a = 0.376 \,\mathrm{m}$ , are used respectively for the ambient hydro-

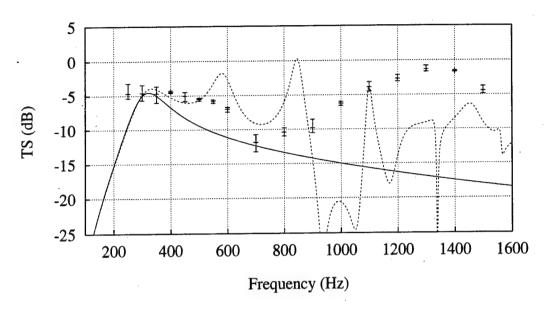


Figure 5.17: Backscatter TS measurements (rms voltage method) performed during the Lake Seneca experiment and comparison with the model. The curves are calculations based on the non-acoustically determined parameters obtained in Chapter 4.3.

static pressure, the sound speed of the water surrounding the sphere, the void fraction, and the radius of the spherical bubble cloud. The error—bars at each source driving frequency indicate the maximum and minimum measured target strength for a sequence of three bubble cloud releases as they crossed the axis of the test range (bins 10–13).

The acoustic scattering measurements agree quantitatively with the prediction of the simple model near the monopole resonance peak ( $\approx 300\,\mathrm{Hz}$ ) as shown in Figure 5.17; this agreement of the experimental measurement with the monopole model is indicated in Table 5.4. Furthermore, in Figure 5.17 there is semi-qualitative agreement out to frequencies as high as 800 Hz. However, the measurements and model diverge at the higher frequencies. This disagreement is likely due to shape dependencies and non-uniform bubble density

	Eq. $(6.4)$ $(m=0)$	Eq. (6.5)	Measured
$\Omega_{ m o}$	$326 \pm 25\mathrm{Hz}$	$294 \pm 25\mathrm{Hz}$	$325\pm25\mathrm{Hz}$
$\mathrm{TS}_{f\!f}$	$-2.9 \pm 0.7  \mathrm{dB}$	$-3.0 \pm 0.7 \mathrm{dB}$	$-3.4 \pm 0.9  \mathrm{dB}^a$

<sup>a</sup>The measured value of TS<sub>ff</sub> was obtained at 300 Hz, and not at  $\Omega_0$ 

Table 5.4: Monopole resonance frequency and target strength for the Lake Seneca bubble cloud.

within the Lake Seneca bubble cloud. For the model, the shape of the bubble cloud is taken to be spherical and the bubble population is uniform throughout; at frequencies above 600 to 700 Hz, this cloud is *not* a uniform sphere. It is likely that if the shape of the bubble cloud were more nearly spherical and that if the bubble population were uniform, then better agreement at all frequencies would have been obtained. Furthermore, the data was obtained over a sparse set of driving frequencies and thus one cannot expect to resolve the sharper peaks and valleys in the higher frequency spectrum without resorting to a frequency spectrum analysis (to be addressed in presentation).

However, for frequencies below  $600\,\mathrm{Hz}$  the cloud is acoustically compact (ka < 1) and the trends observed in the Lake Seneca measurements agree semi–qualitatively with those of the partial wave expansion model. The lowest driving frequency was  $250\,\mathrm{Hz}$ , and a sharp decrease in the target strength predicted by the model (i.e., the Rayleigh scattering regime) below the monopole resonance frequency was not observed. This, in part, is due to the decrease in source level with frequency. The decreasing source level results in a lowered signal to noise ratio, as is evident by the nearly flat elapsed–time response (Figure 5.16) at all frequencies below  $550\,\mathrm{Hz}$ , with the exception of those frequencies near resonance. Still another possibility for the sooner than expected

scattering return near the monopole resonance is that the receiver was not in the far field at these low frequencies. The distance between the target and receiver at the backscatter location was  $20.4\,\mathrm{m}$ , and the wavelength is greater than  $2.5\,\mathrm{m}$ .

# Part III

Backcattering from acoustically compact bubble clouds near the sea surface

# Chapter 6

# Acoustic backscattering from compact bubble clouds beneath the ocean surface

### 6.1 Introduction

Recent studies indicate that bubble layers, plumes, and clouds are produced when waves break (wind speed,  $U > 6\,\mathrm{m/s}$ ), and that these assemblages are convected to depth by Langmuir circulation (Thorpe, 1982; Monahan & Niocaill, 1986). A fundamental question to be answered is what role, if any, do bubble assemblages play in the near–surface production and scattering of sound in the low to mid–frequency range (20 Hz to 2 kHz). There exists ample experimental evidence to suggest that mid–frequency ambient noise levels dramatically increase when waves begin to break (Carey & Browning, 1988;

Hollet, 1994) and that sound scattering from the sea surface possesses a characteristic which is different from that expected by Bragg scattering from gravity waves (McDaniel, 1988). The commonly held view is that persistent bubble clouds, plumes or layers brought on by breaking wave activity can act as efficient scatterers of sound. Indeed, recently published works by Henyey (1991), McDonald (1991), Gragg and Wurmser (1993), Carey and Roy (1993), Gilbert (1993), Sarkar and Prosperetti (1993) indicate a significant role for bubble assemblages in surface backscatter.

In response to these observations a hypothesis has been stated that if bubble clouds and plumes with void fractions,  $\beta$ , ranging from  $10^{-5}$  to  $10^{-2}$  are entrained by surface wave activity, then low-frequency radiated noise can be produced and scattering can occur with little Doppler shift and ample Doppler spread. It has been argued by Prosperetti (1988b) and by Carey and Browning (1988) that, at low frequencies, it is not the individual bubbles driven at their natural resonance frequencies that contribute the most to the scattering cross section of a bubble plume. Rather, sound is scattered primarily by the bubble plume itself via a process in which the bubbles pulsate together in a collective mode of oscillation. This hypothesis is based on the classical theory that when a large number of small bubbles occupy an acoustically compact region, coupled oscillations result and the propagation of sound is determined by the properties of the mixture; the free gas establishes compressibility and the water provides inertia. In such a case, a damped resonant oscillation can result in which the eigenfrequency is determined by the phase speed within the "effective" medium (typically less than 500 m/s) and the characteristic length scale of the plume. Ample evidence of this phenomena, which was originally proposed by d'Agostino and Brennen (1983) has been obtained from the study of low–frequency noise produced by steady–state bubble columns (Yoon et al., 1991) as well as transient bubble plumes (Kolaini et al., 1993; Carey et al., 1993).

# 6.1.1 Prominent, near-surface scattering features

Recent tests of low-frequency active sonar systems (CST-4)<sup>1</sup> indicate the presence of prominent, near-surface targets during conditions of high sea state (Gauss *et al.*, 1993). Those investigators concluded that prominent returns in the 800–1200 Hz range were due to bubble clouds with lifetimes less than 45 seconds. Similar results were obtained at CST-7 by Adair *et al.* (1992), who observed strong returns (target strength > -5 dB) from seemingly short lived targets positioned within a few meters of the surface. Henyey (1991) and McDonald (1991) have developed scattering models based on the application of weak scattering theory to so-called tenuous bubble clouds ( $\beta$  order  $10^{-8}$ ). However, tenuous clouds should persist for more than 45 seconds and are probably not the source of these bright targets.

It is more likely that these short–lived, near–surface targets are the result of bubble plumes generated beneath plunging breakers. Such plumes possess length scales of up to 1 m, void fractions on the order of 10<sup>-3</sup>, and lifetimes on the order of seconds (Monahan, 1988). If the plumes detach, a three–dimensional, high void fraction bubble cloud is momentarily submerged

 $<sup>^{1}\</sup>mathrm{CST}=\mathrm{Critical}$  Sea Test, a research program sponsored by the Office of Naval Research.

at a depth of order 1 meter or possibly more. The contention here is that such a cloud could exhibit resonance scattering properties that translate to high target strengths at low frequencies for short periods of time.

Since the burden of proof lies with the experimentalist, a lake test was carried out in the fall of 1991. (See Chapters 4 and 5 for a detailed description of this effort.) The rational behind the experiment was to measure the frequency–dependent backscatter target strength ( $\mathrm{TS}_{ff}$ ) of a submerged bubble cloud of known properties in the free field under known propagation conditions. The test, which was performed at the NUWC<sup>2</sup> Seneca Lake Sonar Test Facility, involved the production of a bubble cloud of known geometry and void fraction 90 m beneath the surface of the lake. Using a parametric sound source, the target strength of the cloud was measured for frequencies ranging from 250 Hz to over 10 kHz. The test results indicated the presence of a prominent scattering resonance whose level (TS  $\approx -3 \, \mathrm{dB}$ ) and frequency ( $\Omega_0 \approx 325 \, \mathrm{Hz}$ ) are well predicted using classical scattering theory coupled with an effective medium approximation and subject to a priori knowledge of the bubble cloud properties. This theory, which is reviewed below, assumes monopole scattering and free-field conditions. The generalization of this model to near-surface conditions is the topic of this chapter.

<sup>&</sup>lt;sup>2</sup>Naval Undersea Warfare Center

## 6.2 Theoretical model

# 6.2.1 Scattering from bubble clouds in the free field

Here, the usual assumptions are taken in which plane waves are incident on a compliant sphere of radius a surrounded by a continuous medium of density  $\rho$  and sound speed c respectively. Likewise, the sphere is considered to be a homogeneous medium of density  $\rho_e$  and sound speed  $c_e$  respectively. The sphere (bubble cloud) is composed of many air-filled bubbles; thus, the medium is dispersive with an effective density and wave number given by (Commander & Prosperetti, 1989),

$$\rho_e \equiv (1 - \beta)\rho + \beta\rho_g, \tag{6.1}$$

$$k_e^2 = \frac{\omega^2}{c^2} + \frac{4\pi \eta R_o \omega^2}{\omega_o^2 - \omega^2 + 2ib\omega},$$
 (6.2)

where  $k_e$  is the wave number in the liquid,  $R_o$  is the radius of individual bubbles (considered to monodispersed in size),  $\eta$  is the number of bubbles per unit volume,  $\omega_0$  is the angular resonance frequency of the bubbles,  $\beta$  is the void fraction, and b is the complex frequency dependent damping constant. For  $\omega \ll \omega_0$ , one can show that the real portion of the complex phase speed in the mixture is given by,

$$c_e = \sqrt{\gamma P_{\infty}/\beta \rho} \tag{6.3}$$

Here,  $P_o$  is the ambient pressure and  $\gamma$  is the ratio of specific heats of the air. Note that  $c_e$  decreases with increasing void fraction. Indeed, for  $\beta$  greater than approximately  $10^{-2}$ , the effective sound speed of the mixture is less than that encountered in either air or water alone.

For the scattering problem in an infinite medium the Helmholtz equation is solved subject to the boundary conditions of continuous pressure and normal velocity. The far-field solution is a superposition of incident plane waves and scattered spherical waves, the latter of which are represented by an expansion in Legendre polynomials given by

$$p_s(\mathbf{r}) \longrightarrow i \frac{p_o}{2} \frac{e^{ikr}}{kr} \sum_{m=0}^{\infty} (2m+1) \mathcal{R}_m P_m (\cos \theta).$$
 (6.4)

The coefficients  $\mathcal{R}_m$  in Eq. (6.4) satisfy the boundary conditions and describe the reflectivity of the sphere. For acoustically compact scatterers, where  $ka \ll 1$ , the monopole term in the expansion dominates and the far-field backscattered pressure amplitude is approximated by

$$|p_s| \approx p_o \frac{(ka)^2}{3} \left| \frac{1 - gh^2}{\left(gh^2 - \frac{(ka)^2}{3}\right) - i\frac{(ka)^3}{3}} \right|,$$
 (6.5)

where  $g = \rho/\rho_e$ , and  $h = c/c_e$ .

The second term in the denominator accounts for radiation damping while the first term, when set equal to zero, defines a resonance condition. This results in a monopole resonance frequency and scattered pressure amplitude given by

$$\Omega_0 = \frac{1}{2\pi a} \sqrt{\frac{3\gamma P_{\infty}}{\beta (1-\beta)\rho}}.$$
 (6.6)

Note that the expression for  $\Omega_0$  has the same form as the Minneart expression (Eq. 2.21) defining the monopole resonance frequency of a spherical bubble; here  $\beta$  accounts for the velocity defect associated with the two-phase mixture. To obtain the free-field backscattered target strength one can substitute either Eq. (6.4) or, for low-frequency scattering, Eq. (6.5) into

$$TS_{ff} = 20 \log_{10} \left| \frac{p_s}{p_o} \right|_{r=1 \text{ m}}$$
 (6.7)

In Table 5.4 the predicted and resonance  $TS_{f\!f}$  are compared with the measurements obtained at Lake Seneca. Since the concern here is with the monopole resonance response, only the m=0 term was calculated in the expansion. The good agreement between theory and measurement supports the view that an effective media approximation is a reasonable way to model low–frequency bubble cloud scattering; indeed, it appears that one can predict the resonance using the simple algebraic expressions given in Eq. (6.5). It is important to note that the model and the measurements agree despite the fact that the Seneca cloud was spheroidal rather than spherical. This is because a monopole response corresponds to a volume–dependent, breathing mode oscillation. For an acoustically compact scatterer, the monopole resonance behavior is well approximated by a sphere of equal volume. In the case of the Lake Seneca cloud, this corresponds to a sphere  $\approx 0.38 \,\mathrm{m}$  in radius.

# 6.2.2 Scattering from bubble clouds near the ocean surface

At low frequencies we chose to model the acoustically compact target as a monopole scatterer positioned a distance d beneath a plane horizontal surface characterized by a reflectivity coefficient  $\mu$  which can vary between -1 and 0. The primary assumptions here are incident plane waves and far–field propagation. A standard application of the method of images yields the following expression for the near–surface, backscatter target strength:

$$TS_{ns} = TS_{ff} + 10 \log |1 + \mu^2 + 2\mu - 4\mu \sin^2(kd \sin \phi_g)|^2, \qquad (6.8)$$

where  $\phi_g$  is the grazing angle measured relative to the horizontal. The reflection coefficient describes the roughness of the sea surface;  $\mu = -1$  corresponds to a smooth pressure release surface and  $\mu = 0$  corresponds to an extremely rough surface (effectively equivalent to an unbounded medium).

### 6.2.3 The method of images

The solution in Eq. (6.8) is derived using the method of images. Consider the scattering geometry of Figure 6.1. Here, acoustic waves of frequency  $\omega$  and pressure amplitude  $p_i$  are incident on the sea surface at grazing angle  $\phi_g$ . The incident waves are generated by the source S which is located at point S = (-x, -y, -z). The surface can be smooth and pressure release, in which case the amplitude reflection coefficient is  $\mu = -1$ , or it can be roughened by wave action, effectively lowering the magnitude of the reflection coefficient as

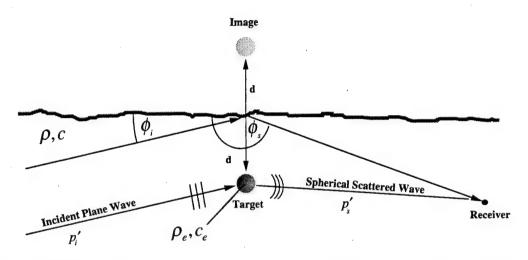


Figure 6.1: Near surface scattering geometry. It is important to note that for backscattering  $\theta_i = \theta_s \equiv \phi_g$ .

alluded to above (this will be discussed more formally in Section 6.2.4).

The 2-D scattering geometry is given in Fig. 6.1, where the z-axis is the vertical direction, and the x-axis denotes the horizontal range. We assume a plane wave incident on a monopole point scatterer a distance d beneath a surface of reflectivity  $\mu$ . For far-field measurements, a monopole scatterer (kd < 1) near a reflecting surface behaves like an acoustic dipole (applying the method of images), thus:

$$P_{inc}(\mathbf{r}) = p_i \left[ 1 + \mu e^{-i2kd\sin\phi_g} \right] e^{i(kx\cos\phi_g + kd\sin\phi_g)}, \tag{6.9}$$

$$P_{scatt}(\mathbf{r}) = p_s \left[ 1 + \mu e^{+i2kd\sin\phi_g} \right] e^{-i(kx\cos\phi_g + kd\sin\phi_g)}, \qquad (6.10)$$

$$\left| \frac{P_{scatt}}{p_i} \right| = \mathcal{F} \left| 1 + \mu^2 + 2\mu \cos\left(2kd\sin\phi_g\right) \right|. \tag{6.11}$$

Here,  $P_{inc}$  and  $P_{scatt}$  refer to the total incident/scattered fields (actual plus image), and  $p_i$  and  $p_s$  are the magnitudes of the incident/scattered signal in the absence of a reflecting surface. The ratio  $p_s/p_i$  on the right side of Eq. (6.11) is the backscatter form function, as derived previously, and should be replaced with Eq. (3.14), or its asymptotic form, Eq. (3.30). However, the data will be presented as TS, so that:

$$TS_{ns} = TS_{ff} + 10 \log |1 + \mu^2 + 2\mu - 4\mu \sin^2 (kd \sin \phi_g)|^2, \qquad (6.12)$$

where the second term on the right is generally referred to as the transmission anomaly (TA)(Urick, 1967; Kinsler et al., 1982). It is easy to show that for  $\mu \approx -1$ , and for clouds that are acoustically close to the surface  $(kd \ll 1)$ , that the argument of the TA can be simplified to  $((kd)^4 \sin^4 \phi_g)$ , which suggests a dipole interference pattern. Thus, high void fraction clouds, which exist only very near the sea surface and have significant low-frequency TS's in the free-field, may not be acoustically "important" due to the effect of surface dipole cancellation. On the other hand, for deeper scatterers, the volume fractions are necessarily lower (by design of nature) and may not resonate at all. It seems plausible that an optimum combination of depth, void fractions and cloud sizes may exist; a notion that we are currently exploring.

# 6.2.4 The Kirchoff approximation

The Kirchoff approximation given by (Urick, 1967; Thorsos, 1990) relates the reflection coefficient to the windspeed, U, as follows:

$$\mu = -e^{-2(kh\sin\phi_g)^2},\tag{6.13}$$

where,

$$h^2 = \frac{\alpha_1 U^4}{4\alpha_2 g^2} \tag{6.14}$$

is the rms–squared wave height predicted by the "Pierson-Moskowitz" wave-height frequency spectra(Pierson & Moskowitz, 1964). Here, U is the wind speed measured 10 m above the sea surface,  $\alpha$ 's are empirical constants, and  $g=9.8~\mathrm{m/s^2}$ . The range of the reflection coefficient is  $-1<\mu<0$ .  $\mu=-1$  corresponds to a smooth pressure release surface (i.e., low frequencies and/or low wind speeds). Similarly, at high frequencies or high wind speeds the surface appears acoustically rough, which disrupts coherent surface reflection yielding free–field acoustic behavior (i.e.,  $\mu \to 0$ ).

It is evident from Eq. (6.5), (6.8), and (6.13) that the resonance backscatter target strength for a near–surface bubble cloud is a complex function of the various parameters. From Eq. (6.5) it is clear that increasing either a or  $\beta$  reduces  $\Omega_0$  and increases the TS<sub>ff</sub>. However, the proximity of the surface is parameterized by kd, i.e., depth is relative to the acoustic wavelength. For  $kd \ll 1$  and  $\mu = -1$ , the scattered pressure scales with  $(d/\lambda)^4$ . From this it

is clear that, although large, high– $\beta$  clouds possess significant cross sections at resonance, they are also subject to substantial dipole interference owing to their close proximity (both oceanographically and acoustically) to the surface. It would appear that there are optimum combinations of physical and environmental parameters leading to significant target strengths (say, larger than -10 dB).

### 6.3 Results and discussion

At this point a "case study" is considered, in which the scattering characteristics of a 5 cm radius cloud of very small bubbles is explored. The grazing angle is fixed at  $10^{\circ}$ . Within the bounds of this simple model, the remaining parameters influencing the scattering characteristics of this cloud are the void fraction, depth, and windspeed. Both free-field and near-surface scattering will be considered. All calculations employ only the m=0 term from Eq. (6.4), thus only the low-frequency (i.e., monopole) backscattering behavior will be considered.

Shown in Figure 6.2(a) is the target strength versus frequency for this cloud resident at a depth of 2m and possessing a void fraction of  $\beta = 5 \times 10^{-3}$ . Note the monopole resonance peak at 786 Hz, where TS<sub>ff</sub> is -9.5 dB. A windspeed of 20 m/s does little to modify the response, for the surface is very rough. It is interesting to note, however, that lowering the windspeed to 7 m/s increases the target strength to +1.95 dB. This is due to "constructive" interference between the real and image scatterers. To see this more clearly,

consider the plot of resonance versus depth show in Figure 6.2(b). The "Lloyds Mirror" interference pattern is clearly the cause of the elevated target strength.

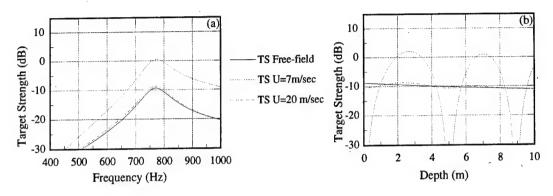
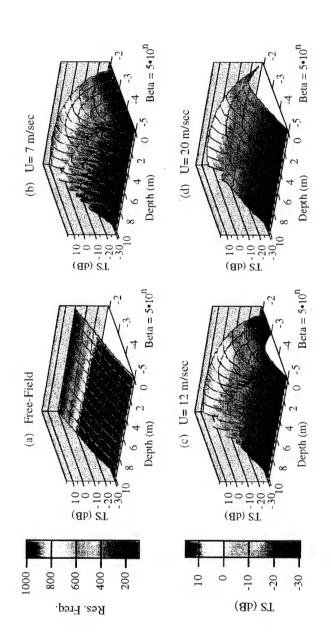


Figure 6.2: Low–frequency backscatter from 5 cm radius cloud for  $\beta=5\times 10^{-3}$  and  $\phi_g=10^\circ$ : (a) depth = 2.0 m; (b) frequency = 786 Hz.

The global scattering properties of this cloud are illustrated in the surface plots of Figure 6.3. Figure 6.3(a) is a plot of the TS at resonance versus depth and void fraction. As predicted in Eq. (6.5), increasing the void fraction results in a significant increase in TS and decrease in  $\Omega_0$ . There is also a weak dependence on submerged depth, as evidence in Eq. (6.3). Figures 6.3(b) through (d) depict the influence of the surface. The surface–dipole interference pattern is most evident at  $7 \,\mathrm{m/s}$ , where over the entire frequency range. Target strengths as high as  $+10 \,\mathrm{dB}$  are observed for clouds with void fraction  $\beta = 5 \times 10^{-3}$  positioned at depths ranging from 7–8 m. However, such clouds are not likely to exist in the open ocean.

A sequential consideration of Figs. 6.3(b) through (d) serves to illustrate the effect of increasing windspeed. As the wind speed increases, the surface become rougher and therefore less influential. Indeed, it appears that at 20 m/s free–field conditions prevail for all but the lowest resonance frequencies (i.e.,



**Figure 6.3:** Monopole resonance backscatter from 5 cm radius cloud for  $\phi_g = 10^\circ$ . The color scale denotes (a): monopole resonance frequency; (b) through (d): TS at resonance.

highest void fractions). The notion of an optimal combination of void fraction and depth is clearly evident in these plots, which exhibit pronounced peaks and valleys as these parameters vary.

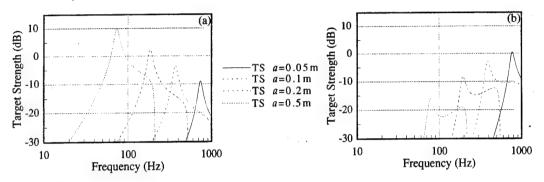


Figure 6.4: Low-frequency backscatter from various clouds for  $\beta = 5 \times 10^{-3}$  and  $\phi_g = 10^{\circ}$ : (a) free-field calculations; (b) depth = 2.0 m and U = 7 m/s.

Now the role played by increasing the size of the bubble cloud is considered. Figure 6.4(a) shows a plot of  $TS_{ff}$  versus frequency for clouds with radii ranging from 5 to 50 cm. All four clouds possess a void fraction of  $\beta=5\times 10^{-3}$ . As expected, increasing cloud size equates to decreasing resonance frequency and increasing target strength in the free field. Conversely, Figure 6.4(b) illustrates the backscatter  $TS_{ns}$  for the same clouds at a depth of 2 m and a surface grazing angle of 10°. Now, the smaller clouds exhibit the higher target strengths at resonance. In general, this is due to the fact that these smaller clouds resonate at higher frequency, which means that the surface is further removed (acoustically speaking) and therefore less influential. However, in this particular instance, the contribution of constructive interference between the real and image scatterers plays a role as well. Regardless, one should not be too quick to ascribe free–field scattering properties to bubble clouds resident near the surface. In some instances, it may be the smallest clouds that make

to largest contributions.

# 6.4 Summary and conclusions

In this study, a simple model describing the low–frequency scattering properties of high void fraction bubble clouds in both the free field and near the ocean surface was developed. This model, which is based on an effective medium approximation and acoustically compact scatters, successfully predicts the results of the bubble cloud scattering experiment carried out at Lake Seneca (Roy et al., 1992). The introduction of the surface is facilitated by the method of images and is subject to the same constraint of low–acoustic frequency imposed by the compact scatterer assumption. This model is not intended to serve as a exact replicate of oceanic bubble cloud scattering. Such an endeavor offers limited utility, for the accuracy of any model is ultimately bounded by the quality of the input data. The model herein was kept simple by design, for only then can the complex physical behavior be expressed in a simple analytical form. Simple, analytic theories facilitate the exploration of parameter space, and more importantly serve to illuminate the underlying physics.

The bubble cloud appears to resonate somewhat like a free bubble (Eq. 6.5). Increasing the size of the cloud decreases the resonance frequency and increases the free–field target strength at resonance. Increasing the void fraction has the same effect due to the reduction in sound speed within the cloud. For example, a 0.5 m radius cloud ( $\beta = 5 \times 10^{-3}$ ) resonates at 75 Hz

and possesses a free field target strength of  $+10\,\mathrm{dB}$  at resonance, whereas a  $0.1\,\mathrm{m}$  radius cloud with the same void fraction resonates at 370 Hz and has a target strength of  $-3.5\,\mathrm{dB}$ .

Introduction of the surface leads to a dipole interaction. For a perfectly smooth surface, the target strength can either increase or decrease depending on the frequency and the depth. Since the resonance frequency is also a function of cloud size and void fraction, the manifold of possible behavior is quite broad. In general, larger clouds are more effected due to the fact that they resonate at lower frequencies; the proximity of the surface is defined relative to the acoustic wavelength. Incoherent scattering brought on by surface roughness reduces the impact of dipole interaction. As the wind picks up, or as the acoustic frequency increases, the environment asymptotes to that of a free field. It is interesting to note that the same environmental conditions that lead to a diminishment of surface dipole interaction also result in high void fraction bubble clouds entrained by breaking waves.

A case study of a 5 cm radius cloud revealed target strengths as high as +10 dB at depths exceeding 7 m and for a windspeed of 7 m/s. For the case of a void fraction cloud at a depth of 2 m, the calculated resonance frequency and target strength was 786 Hz and 0 dB respectively. It is important to recognize that the notion of such a cloud temporarily residing at a depth of 2 meters beneath a breaking wave event is well within the bounds of reason. A 0 dB target strength is quite significant, and we conclude that these small, high void fraction clouds may indeed be the source of the short-lived, near-surface scattering features observed by investigators such as Gauss et al. (1993) and

Adair et al. (1992).

# Chapter 7

# Concluding remarks

The presence of bubbles in water dramatically effects the propagation of sound waves. When a bubble is formed (or "birthed"), it radiates a characteristic damped harmonic acoustic signature at the well known Minneart (monopole) resonance frequency given by  $f_0 = (1/2\pi R_o) \sqrt{3\gamma P_\infty/\rho}$ . In addition to radiating sound, a bubble is an excellent absorber and scatterer of sound as well. In sufficient concentrations, a mixture bubbles in water effectively lowers or raises the sound velocity depending on the acoustic frequency and the sizes of the bubbles.

Substantial laboratory and field evidence suggests that high-void fraction bubble clouds radiate a sound when produced. The acoustic signature, however, is not limited to the relatively high bubble resonance frequencies. Recently, Hollet (1994) and Farmer and Vagle (1989) observed a significant contribution of low-frequency sound (< 1 kHz) coincident with the production of a bubble cloud in the ocean. In addition, many laboratory and well

controlled field experimental efforts have established that bubble cloud production is a viable source of low frequency sound in the ocean (Roy et al., 1992; Carey et al., 1993; Kolaini et al., 1994; Yoon et al., 1991; Nicholas et al., to be submitted) and is in good agreement with the simple theoretical models (Carey & Fitzgerald, 1993; Lu et al., 1990).

Interestingly enough, prior to 1985 there was little or no mention of the possibility that bubble clouds contributed to ambient noise in the ocean below 2 kHz. Currently, this idea is widely accepted. The recent field measurements of Hollet and of Farmer and Vagle, when considered along with the good agreement between the theory and results of Carey's tipping—trough and Kolaini's bucket—drop experiments suggest that the bubble clouds generated beneath breaking waves are a fundamental contributing source of natural low frequency noise in the ocean.

Since a good radiator of sound is a good scatterer of sound, it seems plausible that acoustically compact bubbly regions (clouds) can behave as scatters of sound. The application of a classical theory coupled with the homogeneous mixture assumption based on the low frequency air—water mixture properties produces an analogous single—bubble description of the bubble cloud in which the resonance frequency of the volume mode is described by a modified Minneart formula (Eq. 3.32) and a simplified monopole expression exists for the backscatter target strength (Eq. 3.33). Bubble clouds produced near the sea surface are important in both the production and scattering of low frequency sound.

## 7.1 Collective bubble oscillations

The propagation of acoustic waves in fluids is highly influenced by properties such as absorption and scattering. In fact, for a two-phase fluid consisting of water mixed with a sufficiently large concentration of air bubbles, the sound velocity is frequency dependent. This dependence is a consequence of the marked change of the fluid's natural compressibility when mixed with bubbles (Brekhovskikh & Lysanov, 1991). For frequencies below the fundamental resonance of the individual bubbles, the sound speed can be lowered dramatically. This dependence is solely on the fractional volume or void fraction of bubbles in the fluid (Wood, 1941). The physics behind the lowering of the sound speed is that the bubbly region possesses a density close to that of the liquid, but a much greater compressibility. The free gas establishes the compressibility, while the water provides inertia. If one recalls that the sound speed in a fluid medium is proportional to the square-root of the inverse product of the these quantities, the physical basis for the sound speed defect becomes obvious.

For the low-frequency limiting case, the speed of sound in the mixture is primarily governed by the void fraction  $\beta$ . For  $\beta$  in the range  $10^{-4}$  to  $10^{-2}$ , the mixture sound speed can be lowered to values that fall well beneath the sound speed in the liquid, and even to values below the speed of sound in the gas. This indeed is remarkable! It was demonstrated that the real portion of the complex phase speed in the mixture is given by,

$$c_e = \sqrt{\gamma P_{\infty}/\beta \rho} \tag{7.1}$$

for frequencies below bubble resonance. Here,  $P_o$  is the ambient pressure and  $\gamma$  is the ratio of specific heats of the air. Note that  $c_e$  decreases with increasing void fraction. For  $\beta$  greater than approximately  $10^{-2}$ , the effective sound speed of the mixture is less than that encountered in either air or water alone.

Compared to a bubble, the wavelength of the sound at low frequencies is, by definition, much greater than the bubble radii (and their respective spacing), and thus a bubbly mixture can be considered an effective fluid having a density near that of water and a sound speed that is dependent only on the void fraction and not on the details of the bubble population statistics.

# 7.2 Scattering

In Chapter 3 the classical theory for acoustic scattering from a compliant sphere in the free-field was presented. The solution for the scattered pressure at every point in the sphere's exterior was given by a normal mode expansion (Eq. 3.12). The solution was obtained by simply meeting the requirements on the boundary of the sphere of continuous pressure and continuous normal particle velocity and by exploiting the symmetry of the problem. Although not limited to scattering from acoustically compact bubble clouds, this problem yields itself to a simple and straightforward analysis using the well established collective oscillations model for the effective sound speed and effective density as outlined in Chapter 2.

In addition to the full wave expansion, simple analytical expressions were derived which approximate the monopole resonance frequency  $(\Omega_0)$  and

the target strength (TS) at this resonance of a bubble filled sphere (see Chapter 3.4). It was determined that the monopole resonance frequency and TS depend only on two physical characteristics of the bubble cloud: the total volume and the void fraction ( $\beta$ ). It is important to stress that the monopole TS level and  $\Omega_0$  can always be obtained without a detailed knowledge of the bubble size distribution so long as  $\beta$  and the cloud dimensions are known.

# 7.3 The burden of proof

Since the burden of proof lies with the experimentalist, a lake test was carried out in the fall of 1991 (See Chapters 4 and 5 for a detailed description of this effort). The rational behind the experiment was to measure the frequency—dependent backscatter target strength (TS<sub>ff</sub>) of a submerged bubble cloud of known properties in the free field under known propagation conditions. The test, which was performed at the NUWC¹ Seneca Lake Sonar Test Facility, involved the production of a bubble cloud of known geometry and void fraction 90 m beneath the surface of the lake. Using a parametric sound source, the target strength of the cloud was measured for frequencies ranging from 250 Hz to over 10 kHz. The test results indicated the presence of a prominent scattering resonance whose level (TS  $\approx$  -3 dB) and frequency ( $\Omega_0 \approx 325$  Hz) are well predicted using classical scattering theory coupled with an effective medium approximation and subject to a priori knowledge of the bubble cloud properties. This theory, which is described in Chapters 2 and 3, assumes monopole scattering and free—field conditions, and models the acoustically compact cloud

<sup>&</sup>lt;sup>1</sup>Naval Undersea Warfare Center

as a sphere.

The acoustically compact cloud produced during the Lake Seneca experiment possess higher—order poles which are not easily described by the simple analytical approach outlined in this dissertation. This is presumably due to the fact that the cloud was neither a perfect sphere nor a perfect cylinder. Unlike the monopole mode, the higher order resonances are dependent on the shape of the cloud and the localized void fraction distribution within the cloud.

Based on the relatively good theoretical agreement with experimental findings of the Lake Seneca experiment (and recent ambient noise experiments), it is quite plausible that bubble clouds are an important mechanism related to the increase in surface scattering levels observed in the ocean during periods of high sea state and for frequencies between 20 and 2000 Hz. In the free–field, an acoustically compact bubble cloud scatters sound effectively in its monopole mode and in quantitative agreement with the model based on a spherical bubble cloud of uniform void fraction. To our knowledge, this is the first report of monopole resonance scattering from a bubble cloud.

### 7.4 Scatter from subsurface bubble clouds

Although detailed experimental studies of the frequency dependent acoustic scattering from individual bubble clouds immediately beneath the sea surface have not been conducted, the simple model presented in Chapter 6 (Lloyd's Mirror) suggests that near–surface bubble clouds are strong scattering targets, even at the low frequencies. The most important determining factor in this

case is the monopole resonance frequency of the cloud (dependent on  $\beta$  and a) and the cloud depth beneath the surface relative to the wavelength; for small  $d/\lambda$  or small grazing angle, the TS is proportional to  $(d/\lambda)^4$ . A monopole bubble cloud scatterer beneath the pressure release sea surface is acoustically coupled to the surface and behaves as a simple dipole scatterer (method of images).

In modeling the acoustic scatter from bubble clouds beneath the sea surface, we are presented with an optimization problem which can be summarized as follows: The free-field scattering model suggests that the monopole resonance frequency decreases and the TS of the cloud increases as the cloud radius is increased (all other parameters held fixed). Near the sea surface, however, a cloud having a lower resonance frequency (and thus larger TS) is less important due to destructive dipole interference. Furthermore, a cloud with a low resonance frequency (say  $\Omega_0 = 100\,\mathrm{Hz}$  or  $\lambda = 15\,\mathrm{m}$ ) is physically unlikely to penetrate to depths in which the destructive dipole interference is overcome. These high void fraction clouds are more likely to be present at high sea states. Moreover, as the sea state becomes rougher, the dipole interference effects become less important since the reflectivity of the sea surface decreases as it diverges from its smooth pressure release value. Thus, the larger or higher void fraction clouds play an increasing larger role as the sea state increases.

It should be noted that as with the Chapman–Harris and Chapman–Scott sea surface scattering results and the associated empirical curves, any surface scattering model will likely display a "dipole–like" dependence similar to that illustrated in Chapter 6 (i.e.,  $(d/\lambda)^4 \sin^4 \phi_g$ ). However, the hypothesis

that relatively densely populated bubble clouds are the source of the scattered sound is on an especially strong physical footing since the low–frequency sea surface scattering levels are observed to increase dramatically at the moment waves begin to break. Indeed, as is the case with underwater ambient noise production, indications are that two distinct mechanisms are likely responsible for the anomalous scattering results—one prior to, and one after wave breaking. Finally, the recent surface scattering results reported by Adair and Huster indicate that transient, spatially discrete, high target strength scattering centers exist in the upper 5 m of the ocean surface during periods of high sea state. These factors are not only strongly suggestive of bubble clouds, but are also consistent with the simple model described in Chapter 6.

### 7.5 Remarks

The bubble cloud appears to resonate somewhat like a free bubble. Increasing the size of the cloud decreases the resonance frequency and increases the free-field target strength at resonance. Increasing the void fraction has the same effect due to the reduction in sound speed within the cloud. For example, a  $0.5\,\mathrm{m}$  radius cloud ( $\beta=5\times10^{-3}$ ) resonates at 75 Hz and possesses a free field target strength of  $+10\,\mathrm{dB}$  at resonance, whereas a  $0.1\,\mathrm{m}$  radius cloud with the same void fraction resonates at 370 Hz and has a target strength of  $-3.5\,\mathrm{dB}$ .

Introduction of the surface leads to a dipole interaction. For a perfectly smooth surface, the target strength can either increase or decrease depending on the frequency and the depth. Since the resonance frequency is also a function of cloud size and void fraction, the manifold of possible behavior is quite broad. In general, larger clouds are more effected due to the fact that they resonate at lower frequencies; the proximity of the surface is defined relative to the acoustic wavelength. Incoherent scattering brought on by surface roughness reduces the impact of dipole interaction. As the wind picks up, or as the acoustic frequency increases, the environment asymptotes to that of a free field. It is interesting to note that the same environmental conditions that lead to a diminishment of surface dipole interaction also result in high void fraction bubble clouds entrained by breaking waves.

In this study, a simple model describing the low–frequency scattering properties of high void fraction bubble clouds in both the free field and near the ocean surface was developed. This model, which is based on an effective medium approximation and acoustically compact scatters, successfully predicts the results of the bubble cloud scattering experiment carried out at Lake Seneca for frequencies consistent with the model assumptions (Roy et al., 1992). The introduction of the surface is facilitated by the method of images and is subject to the same constraint of low–acoustic frequency imposed by the compact scatterer assumption. This model is not intended to serve as an exact replicate of oceanic bubble cloud scattering. Such an endeavor offers limited utility, for the accuracy of any model is ultimately bounded by the quality of the input data. The model herein was kept simple by design, for only then can the complex physical behavior be expressed in a simple analytical form. Simple, analytic theories facilitate the exploration of parameter space, and more importantly serve to illuminate the underlying physics.

# Appendix A

# Target Strength

# A.1 Definition of target strength

Measurement of the scattering of acoustic waves from a submerged target is a complex process for which one does not always know the shape, volume, orientation, or composition of the target. For this reason, the sonar community has defined the relation target strength, which refers to the relative echo strength returned by an underwater target. The targets may be of military interest, such as mines, torpedoes, or submarines; of commercial interest, such as schools of fish sought by fish finding sonar's; or of academic interest, as in the case for a spherical bubble cloud. The target strength, as defined by Urick (1967), is

$$TS = 10 \log_{10} \left(\frac{I_s}{I_i}\right)_{R_{\text{ref}}=1 \text{ m}}, \tag{A.1}$$

where  $I_i$  and  $I_s$  are the incident and scattered acoustic intensities, respectively. Furthermore, these values are evaluated at a reference distance of 1 m from the apparent origin of the scattered sound. Since the acoustic intensity is proportional to the pressure amplitude squared, the TS can be written

$$TS = 20 \log_{10} \frac{|p_s|}{|p_i|_{R_{ref}=1 \text{ m}}}$$
 (A.2)

The reference distance of 1 m is arbitrary and often causes underwater objects to have positive target strength. Positive values should not be interpreted as meaning that the sound returned by the object is greater than that incident upon it; rather it should be regarded as a consequence of an arbitrary reference distance—for example, if a reference distance of 1 km were used, all customary targets would possess negative target strength (due to spherical spreading of he outgoing acoustic wave). If the sonic wavelength inside the scatterer is less than or on the order of the size of the target, resonances can occur, possibly resulting in elevated target strengths.

In Figure A.1, the geometry for a scattering measurement from an arbitrary target is shown. Here, H is the point at which the pressure amplitude of the scattered wave is imagined to be measured and O is the acoustic center of the target. Note that O may lie inside or outside the target itself and is the point from which the returned sound appears to originate on the basis of measurements made at some large distance from the target.

As an illustration, consider the target strength of a perfectly reflecting sphere for large ka. The sphere (radius a) is considered large and rigid and

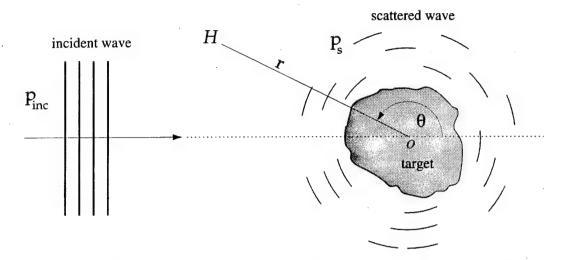


Figure A.1: Geometry of a target strength measurement from an arbitrary scatterer. A plane wave  $(p_{\rm inc})$  is incident on the target from the left. A detector located at point H records the scattered signal  $(p_{\rm s})$  at a distance r and angle  $\theta$  from the forward scattering direction. Here, the origin of the scattered signal is the center of the target.

is insonified by plane waves of intensity  $I_i$ . The acoustic power intercepted by the sphere is  $\mathbb{P}_i = \pi a^2 I_i$ . Sound is scattered in all directions uniformly and hence the total scattered power is  $\mathbb{P}_s = 4\pi r^2 I_s$ . Since rigid targets do not absorb energy, the scattered energy must be equal to that incident on the target. Hence, the scattered acoustic intensity is given by the relation

$$I_s = \frac{\pi a^2 I_i}{4\pi r^2} = I_i \frac{a^2}{4},\tag{A.3}$$

and thus the target strength for a perfectly reflecting sphere is

$$TS = 20 \log \frac{a}{2}.$$
 (A.4)

Evidently then, for a 2 m-radius perfectly reflecting sphere the TS is 0 dB.

Note that for this example, the reference distance  $R_{\text{ref}} = 1 \,\text{m}$  is inside the sphere.

A prerequisite knowledge of the size, shape, and composition of the scatterer are not necessary to perform target strength measurements (only the range and magnitudes of the incident and scattered pressure fields are needed). The preceding statement does not apply when one wishes to compare a TS measurement with a theoretical model—in which case, the size, shape, and composition of the scatterer must be taken into account.

# A.2 TS measurements — Steady state

Scattering experiments are usually conducted using directional transmitters and pulses of sound rather than continuous plane waves. This is done primarily to avoid unwanted reverberation from boundaries (tank walls, surfaces, etc.) and the difficulty of measuring a scattered signal in the presence of the cw incident wave. The most common experimental scattering arrangement is the monostatic (pulse–echo) geometry in which the source and receiver are located at the same or closely spaced points. If the transmitter and receiver are separated by a wide distance, then the configuration is known as bistatic.

Consider Figure A.2 in which the geometry for a bistatic (or in this case, quasi-monostatic) backscattering target strength measurement is illustrated. A pulsed directional source (S) is aimed at the target (T). Both the incident and scattered waveforms are measured at some point between the target and the source using the receiving hydrophone (H). The separation distances be-

tween each of these objects are known accurately. Furthermore, the distances  $R_{\rm SH}$ ,  $R_{\rm ST}$ , and  $R_{\rm TH}$  are large enough that the incident field is sufficiently planar, and the scattered field appears to originate from a point.

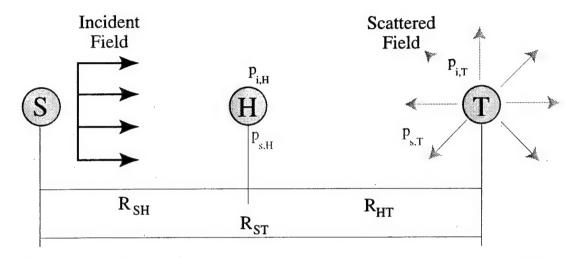


Figure A.2: Geometry for quasi-monostatic scattering measurement. A directional source (S) is aimed at the target (T). Both the incident and scattered waveforms are measured at the receiver (H). The distances between the components are known accurately. The incident pressure amplitude measured at the receiver H is  $p_{i,H}$ . Incidentally, the source and receiver are separated by a large distance so that the incoming waves appear planar. Similarly, the scattered pressure evaluated at unit distance from the target T is  $p_{s,T}$ . H is small enough that it does not interfere with the incident field.

At time  $t_o = 0$ , the source sends out a pulse of sound of duration  $\tau$  and nearly constant angular frequency  $\omega = 2\pi f$ . The ratio of  $\tau$  to the period 1/f should be much larger than 1 in order for the pulse waveform to reach a steady state amplitude. However, the pulse length  $\tau$  must be less than  $2R_{\rm TH}/c$  so that the incident and scattered pulses measured at detector H do not overlap at the detector location (c is the sound speed of the surrounding fluid). At time  $t_1 = R_{\rm SH}/c$ , the incident waveform is received at detector H,

and the steady state voltage amplitude of this main bang pulse is measured (MBV); the pressure amplitude of the incident pulse measured at the receiver is  $p_{i,H}$ .

The incident pressure wave continues to propagate toward the target, intercepting it at time  $t_2 = R_{\rm ST}/c$  at which point its amplitude is  $p_{\rm i,T}$ . The target responds to the incident pressure disturbance by absorbing and scattering sound. The portion of the scattered wave propagating in the backward direction is received by detector H at time  $t_3 = (R_{\rm ST} + R_{\rm TH})/c$ . The echo voltage amplitude recorded is EV; the backscattered pressure amplitude measured is  $p_{\rm s,H}$ .

The target strength of the scatter is given by Eq. (A.2), and thus we are left to determine the scattered and incident pressure amplitudes evaluated at a distance of 1 m from the target origin. These absolute quantities directly relate to the measured voltages after making corrections for receiving amplifier gain, hydrophone calibration coefficients, and transmission losses due to 1/R spherical spreading of the pulses.

The amplitude of the scattered pulse at receiver H is related to the measured voltage amplitude EV as follows:

$$20 \log \frac{p_{s,H}}{p_{ref}} = 20 \log \frac{EV}{V_{ref}} + C - G_{echo}.$$
 (A.5)

where  $p_{ref} = 1\mu \,\mathrm{Pa}$  and  $V_{ref} = 1$  volt by convention. The additive term C is the decibel calibration coefficient of the receiver, and  $G_{\mathrm{echo}}$  is the amplifier gain (in dB) employed to enhance the echo signal.

The echo pressure amplitude evaluated at unit distance from the target  $(p_{s,T})$  is related the measured scattered pressure amplitude at receiver H by

$$20 \log \frac{p_{s,T}}{p_{ref}} = 20 \log \frac{p_{s,H}}{p_{ref}} + 20 \log \frac{R_{TH}}{R_{ref}},$$

$$= 20 \log \frac{EV}{V_{ref}} + C - G_{echo} + 20 \log \frac{R_{TH}}{R_{ref}}, \quad (A.6)$$

where the quantity  $20 \log(R_{\rm SH}/R_{ref})$  is known as the transmission loss due to spherical spreading.

Similarly, the measured incident main bang voltage (MBV) is related to the main bang pressure amplitude  $(p_{i,H})$  at the receiver by

$$20 \log \frac{p_{i,H}}{p_{ref}} = 20 \log \frac{MBV}{V_{ref}} + C - G_{mb}, \tag{A.7}$$

where  $G_{\rm mb}$  is the amplifier gain factor in dB used to enhance the incident signal measurement. Thus, the source level at the S is related to MBV by

$$SL_0 = 20 \log \frac{p_{i,H}}{p_{ref}} + 20 \log \frac{R_{SH}}{R_{ref}},$$
  
 $= 20 \log \frac{MBV}{V_{ref}} + C - G_{mb} + 20 \log \frac{R_{SH}}{R_{ref}},$  (A.8)

after correcting for spherical spreading losses. Using Eq. (A.8), the main bang pressure amplitude evaluated at a unit distance from the target  $(p_{i,T})$  is given in the following expression:

$$20 \log \frac{p_{i,T}}{p_{ref}} = SL_0 - 20 \log \frac{R_{ST}}{R_{ref}}, \tag{A.9}$$

where  $R_{\rm ST}$  is the distance between the source and the target.

Now, the target strength can be evaluated using Eq. (A.2) by substituting Eq. (A.6), (A.8), and (A.9). Hence,

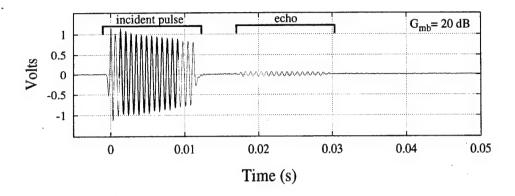
$$TS = 20 \log \left| \frac{p_{s,T}}{p_{i,T}} \right|_{r=1 \text{ m}},$$

$$= 20 \log \frac{\text{EV}}{\text{MBV}} - 20 \log \frac{R_{\text{SH}}}{R_{\text{ST}}R_{\text{TH}}} - (G_{\text{echo}} - G_{\text{mb}}), \tag{A.10}$$

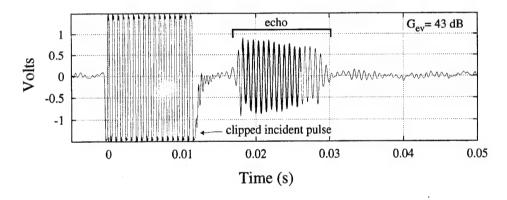
Notice that all of the relative quantities (i.e.,  $V_{ref}$ ,  $R_{ref}$ ,  $p_{ref}$ ) have been eliminated. Furthermore, Eq. (A.10) is independent of the transmitter source level ( $SL_0$ ) and the receiver calibration coefficient (C). Thus for a bistatic backscattering measurement, only the separation distances between the components, the difference in amplifier gain factors, and the voltage amplitudes of the main bang and echo pulses need to be measured in order to accurately determine the target strength of a scatterer at a given source driving frequency.

Shown in Figure A.3 is an example of a steady state TS measurement. The incident and scattered pulse waveforms were measured from a single hydrophone. The source driving frequency is 1400 Hz, and the incident pulse length is 12 ms. The incident signal was amplified by 20 dB. To increase the resolution of the scattered pulse, the echo signal was amplified by 43 dB. The darker region in each plot corresponds to an 8 ms window over which the steady state rms voltage amplitude measurements were conducted. In this case, the rms voltages are MBV =  $0.6\,\mathrm{V_{rms}}$ , EV =  $0.46\,\mathrm{V_{rms}}$ . The total transmission loss factor,  $20\,\mathrm{log}\,R_{\mathrm{SH}}/(R_{\mathrm{ST}}R_{\mathrm{TH}})$ , is -22.38 dB. For the settings listed above, the

measured target strength as given by Eq. (A.10) is  $TS = -2.9 \, dB$ .



(a) Incident waveform



(b) Backscattered waveform

Figure A.3: (a) Incident and scattered signal and (b) same as (a), except added gain to enhance the echo waveform. The signals were acquired from a single hydrophone. The source driving frequency is 1400 Hz, and the incident pulse length is 12 ms. The darker region in each plot corresponds to an 8 ms window over which the steady state rms voltage amplitude measurements were made. In this case, the rms voltages are MBV =  $0.6\,\mathrm{V_{rms}}$ , EV =  $0.46\,\mathrm{V_{rms}}$ . The total transmission loss factor is  $-22.38\,\mathrm{dB}$ . Thus, the measured target strength as given by Eq. (A.10) is -2.9 dB.

#### A.3 TS measurements — transient pulses

In the previous section, the experimental method for determining the backscatter target strength of an arbitrary scatterer versus frequency was outlined. This method requires the use of an incident acoustic pulse at a well defined single frequency, and pulses that are long enough so that a steady state reflected pulse is achieved. In order to determine the target strength over range of frequencies, numerous steady state amplitude measurements must be conducted, one at each frequency of interest.

However, no sound of finite duration produces a pure single frequency note since such a sound theoretically lasts an infinite amount of time. Thus a sound pulse contains frequency components other than the desired frequency which ultimately contribute to the scattered field. In general, the relation between any time-dependent signal  $(p_{inc}(t))$  and the frequency spectrum of the signal is given by Fourier's Integral theorem. That is, the signal can be written in the form

$$p_{\rm inc}(t) = \int_{-\infty}^{\infty} \tilde{p}_{\rm inc}(\omega) e^{i\omega t} d\omega, \qquad (A.11)$$

where  $\tilde{p}_{\rm inc}(\omega)$  is the frequency spectrum of the signal and  $\omega$  is the angular frequency. Likewise, the frequency spectrum can be written as

$$\tilde{p}_{\rm inc}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} p_{\rm inc}(t) e^{-i\omega t} dt.$$
 (A.12)

Eq. (A.11) and (A.12), are just generalizations of the corresponding equations

applicable to Fourier series of periodic functions. The time series and frequency spectrum,  $p_{\text{inc}}(t)$  and  $\tilde{p}_{\text{inc}}(\omega)$ , are generally complex.

Recall from Chapter 3 that the amplitude of the wave scattered from a sphere can be written in terms of a scattering form function  $(\mathcal{F})$  and the incident pressure amplitude  $(p_o)$  as follows:

$$p_s = \frac{a}{2r} p_o \left| \mathcal{F}(ka) \right|, \tag{A.13}$$

where a is the sphere radius, r is the distance from its center to the point of reception, and  $\omega$  is the angular frequency. The amplitude of the incident acoustic pressure measured at the center of the sphere, in its absence, may be regarded as a function of time and is given by

$$p_{\rm inc}(t) = p_o e^{i\omega_o t}. (A.14)$$

Rewriting Eq. (A.13) in terms of the time dependence of Eq. (A.14), the scattered acoustic pressure becomes a function of time:

$$p_s(t) = \frac{a}{2r} |\mathcal{F}(\omega_o)| p_o e^{i\omega_o t}. \tag{A.15}$$

From Eq. (A.11) and (A.12), it is clear that a time-dependent incident pressure pulse,  $p_i(t)$  containing frequencies expressed in  $\omega$  space has a frequency spectrum  $\tilde{p}_i(\omega)$ . Similar expressions exist for the time-dependent scattered acoustic pulse and its frequency spectrum. From Eq. (A.15), the scattered acoustic pressure amplitude at a single frequency, it can be shown

that the relationship between the scattering form function and time-dependent incident and scattered pulses is given by

$$\int_{-\infty}^{\infty} p_s(t) e^{-i\omega' t} dt = \frac{a}{2r} \mathcal{F}(\omega) \int_{-\infty}^{\infty} p_{\text{inc}}(t) e^{-i\omega' t} dt,$$

$$\tilde{p}_s(\omega) = \frac{a}{2r} \mathcal{F}(\omega) \tilde{p}_{\text{inc}}(\omega) \qquad (A.16)$$

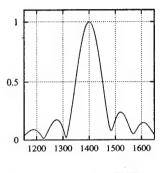
or further,

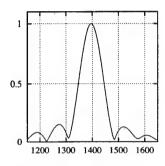
$$|\mathcal{F}(\omega)| = \frac{2r}{a} \frac{|\tilde{p}_s(\omega)|}{|\tilde{p}_{\rm inc}(\omega)|}.$$
 (A.17)

This analysis yields a spectrum of measurements equivalent to the bandwidth of the incident pulse, rather than a single point as in the steady state analysis. That is, using Fourier techniques, one can determine the frequency response over a range of frequencies from a single scattering measurement. This is because the bandwidth of a 'mono-frequency' pulse of center frequency  $f_o$  and pulse length  $\tau$  is approximately  $1/\tau$ . Thus for short pulse lengths (i.e., large bandwidths), a broad spectrum of information can be obtained by comparing the frequency content of the incident and scattered waveforms.

To perform this analysis using experimentally acquired data, the incident and scattered waveforms are digitized and the Fourier transforms calculated using the well known Fast Fourier Transform (FFT) algorithms. Consider the previous example shown in Figure A.3. Peak normalized frequency spectrums of the incident and scattered pulses are shown in Figure A.4. Although the spectrums appear very similar, it is clear that the peak frequency of the

scattered pulse is shifted to a slightly lower value than center frequency of the incident pulse. Also, for frequencies beyond the center frequency, the amplitude of the scattered pulse frequency spectrum falls off more quickly than the incident wave.





(a) Incident pulse FFT.

(b) Scattered pulse FFT

Figure A.4: Fourier transform of incident and scattered waveforms shown in Figure A.3. The driving frequency is 1.4 kHz, and the pulse length is 12 ms. In these figures transform amplitudes are normalized to the peak value at the center frequency.

The target strength can be determined from:

$$\mathrm{TS} = 20 \log \left| \frac{\tilde{p}_{\mathrm{s,T}}(\omega)}{\tilde{p}_{\mathrm{i,T}}(\omega)} \right|_{r=1 \, \mathrm{m}},$$

$$=20\log\frac{\tilde{\mathrm{EV}}(\omega)}{\tilde{\mathrm{MBV}}(\omega)}-20\log\frac{R_{\mathrm{SH}}}{R_{\mathrm{ST}}R_{\mathrm{TH}}}-(G_{\mathrm{echo}}-G_{\mathrm{mb}}). \tag{A.18}$$

Here, the gain, transmission loss, and coefficients (SH, ST, TH) have the same meaning as described in the previous section. Generally, Eq. (A.18) is evaluated only over those frequencies within 3 dB of the of the peak incident signal

level. For the frequency spectrums in Figure A.4, the bandwidth of the pulses is 83 Hz, and the resulting target strength decreases nearly linearly with frequency resulting in target strengths between -0.3 and -2.5 dB, very close the to the steady state measurement of -2.9 dB. The most attractive advantage of using Fourier analysis, compared to the steady state measurement techniques, is that a wider spectrum of data is obtained from a single measurement. Therefore, fewer measurements are required to obtain the same spectrum. In addition, this technique offers improved frequency resolution and is thus better equipped to resolve sharp resonance peaks in the measured form function.

## Appendix B

## Seneca-2 test schedule and recording parameters

Table B.1: Seneca-2 Data Runs and Recording Parameters.

Dat	ta Run	log		Tape Desc	Fil	ters	Rec	cording Channe	l Description	
FQ.	Cloud	pg	Spd	Count	нР	LP	Ch 1	Ch 2	Ch 3	Ch 4
			38	1:330.0—340.0	100	7000	41 dB, H1	41 dB, H2	41 dB, H3	41 dB, H4
6400	MB1	43	30	1:355.0-360.0	100	1000	61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4
	A1			1:370.0-385.0			0.42,	,		
	A2			1:385.0—401.0						
	A3			1:401.0—416.0						
	A4			1:416.0-431.0						
4000	A5	45	38	1:522.0—537.0	100	6000	41 dB, H1	41 dB, H2	41 dB, H3	41 dB, H4
4800	мв	45	36	1:537.0-552.0	100	0000	61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4
	A1	:		1:552.0—567.0			01 42, 111			
	A2			1:567.0—582.0						
	A3			1:582.0-598.0						
	A4	10	- 00	2:020.0—035.0	100	4400	41 dB, H1	41 dB, H2	41 dB, H3	41 dB, H4
3200	MB	46	38		100	4400	61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4
	A1			2:035.0050.4			01 dB, 111	01 02, 112	0142,20	010-, 11
	A2			2:050.4065.6						
	A3			2:065.0—080.8	Ì					
	A4			2:080.8-095.9						
	A5			2:111.1—126.3	100	2000	20 JD U1	38 dB, H2	38 dB, H3	38 dB, H4
.2400	MB	. 48	38	2:217.0-229.0	100	3000	38 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4
	<b>A</b> 1			2:232.0-247.0			61 dB, H1	01 015, 112	01 01, 113	0142, 111
	A2			2:247.0—268.0						
	A3			2:268.0—285.0						
	A4			2:285.0-304.0	100	0000	38 dB, H1	38 dB, H2	38 dB, H3	38 dB, H4
2000	MB1	50	38	2:399.0—418.0	100	2600 2600	61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4
	A1			2:437.0-456.0	100	2600	or ab, iii	0100, 112		01 02,
	A2			2:456.0-475.0			٠			
	A3			2:475.0-494.0						
	A4	-	-	2:494.0-513.0	100	2200	61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4
1600	A1	51	38	2:646.0665.0	100	2200	61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4
	A2			2:665.0—684.0			orub, m	J1 (1D), 112	37 42, 110	22 42, 124
	A3			2:684.0700.0				•		
	A4			3:010.2—029.2					1	
	A5			3:029.2-048.1						
	<b>A</b> 6			3:048.1—067.1	1,,	,,,	Colo Possini			
Hz	#	#	cm/s	<u> </u>	Hz	Hz	Gain, Receiver			

Continued on next.page.

Table B.1: (continued) .

Da	ta Run	log	<u> </u>	Tape Desc	Fil	ters	Recording Channel Description								
FQ	Cloud	pg	Spd	Count	нР	LP	Ch 1	Ch 2	Ch 3	Ch 4					
4000	мв	55	38	4:020.0-038.9	100	1800	43 dB, H1	33 dB, H2	43 dB, H3	33 dB, H4					
4000	A1	00	50	4:057.9—076.8	100	1000	61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4					
	A2			4:076.8-95.0			0142, 111		01 42, 110	0142, 111					
	A3			4:095.0114.0											
	A4			4:114.0—133.0											
	A5			4:133.0153.0											
	A6			4:153.0—171.0											
1200	мв	53	38	3:218.0—237.0	100	1800	43 dB, H1	43 dB, H2	43 dB, H3	43 dB, H4					
1200	A1		00	3:256.0—275.0			61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4					
	A2			3:275.0-294.0			0142,000	01 02, 112	, , , , , , , , , , , , , , , , , , , ,						
	A4			3:313.0-332.0											
	A5			3:332.0-351.0											
7500	MB1	53	38	3:285.0—304.0	100	8200	41 dB, H1	41 dB, H2	41 dB, H3	41 dB, H4					
7300	A1	33	30	3:304.0—323.0	100	3200	61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4					
	A2			3:323.0—341.0			orub, m	01 00, 112	0145, 110	01 02, 114					
	A2			3:342.0-361.0	:		. (								
	A4			3:361.0-380.0											
800	MB	53	38	3:427.0—446.0	100	1000	46 dB, H1	46 dB, H2	46 dB, H3	46 dB, H4					
800	A1	05	00	3:465.0—484.0	100	1000	61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4					
	A2			3:484.0—503.0			0,		,						
	A3			3:503.0—522.0											
	A4			3:522.0—535.0											
11k	мв	58	38	4:494.0-513.0	100	1300	33 dB, H1	33 dB, H2	33 dB, H3	33 dB, H4					
	<b>A</b> 1			4:513.0—532.0			61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4					
	A2			4:532.0—551.0											
	A3			4:551.0569.0											
	A4			4:569.0-588.0											
9000	PSc MB	61	38	5:594.0—617.0	100	12k	-4 dB, H1	-4 dB, H2	-4 dB, H3	-4 dB, H4					
	PSc A1			5:617.0-640.0			21 dB, H1	21 dB, H2	21 dB, H3	21 dB, H4					
	PSc A2			5:640.0—662.0						:					
	PSc A3			5:662.0—686.0											
	PSc A4			5:686.0-710.0											
	PSc A6			6:087.0109.0											
Hz	#	#	cm/s		Hz	Hz	Gain, Receiver								

Continued on next page.

Table B.1: (continued) .

. Da	ta Run	log		Tape Desc	Fil	ters	Rec	cording Channe	l Description	
FQ	Cloud	pg	Spd	Count	нР	LP	Ch 1	Ch 2	Ch 3	Ch 4
10k	PSc MB	61	38	6:223.0—246.0	100	13k	3 dB, H1	3 dB, H2	3 dB, H3	3 dB, H4
	PSc A1			6:246.0-269.0			21 dB, H1	21 dB, H2	21 dB, H3	21 dB, H4
	PSc A2			6:269.0-292.0						
	PSc A3			6:292.0—314.0						
	PSc A4			6:314.0-337.0						
3000	PSc MB	65	38	6:497.0—519.0	100	4500	44 dB, H1	44 dB, H2	44 dB, H3	44 dB, H4
	PSc A1			6:519.0—542.0			61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4
	PSc A2			6:542.0—565.0						
	PSc A3			6:565.0587.0			·		,	
	PSc A4			6:587.0—610.0						
xxx	мв	65	хx	6:632.0—655.0	100	4500	41 dB, H1	41 dB, H2	41 dB, H3	41 dB, H4
	<b>A</b> 1			6:655.0678.0			61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4
	A2			6:678.0—700.0						
14k	PSc MB	65	76	7:045.9—091.5	100	16k	-8 dB, H1	-10 dB, H2	-8 dB, H3	-8 dB, H4
	PSc A1			7:091.5—136.6			11 dB, H1	11 dB, H2	11 dB, H3	11 dB, H4
	PSc A2			7:136.6181.9						
	PSc A3			7:181.9—227.0						
	PSc A4			7:227.0—273.0						
	PSc A5			7:273.0—319.0						
	PSc A6			7:319.0-364.0						
1000	·A1	69	38	7:430.0-452.0	100	2000	63 dB, H1	63 dB,H2	63 dB, H3	63 dB, H4
	A2			7:475.0-498.0			63 dB, H1	63 dB, H2	63 dB, H3	63 dB, H4
	<b>A</b> 3			7:498.0—521.0						
	A4			7:521.0—536.0						
1200	MB	69	76	7:627.0—631.0	100	1800	40 dB, H1	40 dB, H2	40 dB, H3	40 dB, H4
	A1			7:631.0—646.0			63 dB, H1	63 dB, H2	63 dB, H3	63 dB, H4
	A2			7:646.0-661.0						
	A3			7:661.0675.0		,		-		
	A4			8:020.0—035.0						
	<b>A</b> 5			8:035.0—050.0						
	<b>A</b> 6			8:065.0-080.0						
1300	MB	70	38	8:141.0—160.0	100	2000	40 dB, H1	40 dB, H2	40 dB, H3	40 dB, H4
	A1			8:144.0—160.0			63 dB, H1	63 dB, H2	63 dB, H3	63 dB, H4
	A2			8:160.0—175.0						
	<b>A</b> 3			8:175.0—190.0						
	A4			8:190.0-206.0						
Hz	#	#	cm/s		Hz	Hz	Gain, Receiver			l

Continued on next page.

Table B.1: (continued).

Da	ta Run	log		Tape Desc	Fil	ters	Re	cording Channe	el Description	
FQ	Cloud	pg	Spd	Count	HР	LP	Ch 1	Ch 2	Ch 3	Ch 4
1100	МВ	71	38	8:250.0-253.0	100	1800	41 dB, H1	41 dB, H2	41 dB, H3	41 dB, H4
	A1			8:253.0-269.0			63 dB, H1	63 dB, H2	63 dB, H3	63 dB, H4
	A2			8:269.0-284.0						
	A3			8:284.0-299.0						
	A4			8:299.0-314.0						
	A5			8:314.0329.0						
8250	мв	71	38	8:374.0—378.0	100	9500	41 dB, H1	41 dB, H2	41 dB, H3	.41 dB, H4
	A1			8:378.0-393.0			61 dB, H1	61 dB, H2	61 dB, H3	61 dB, H4
	A2			8:393.0-409.0						
	A3			8:409.0—424.0						
	A4			8:424.0-439.0						
	A5			8:439.0—454.0						
7500	MB	72	38	8:469.0—473.0	100	9500	43 dB, H1	43 dB, H2	43 dB, H3	43 dB, H4
	A1			8:473.0—488.0			66 dB, H1	66 dB, H2	66 dB, H3	66 dB, H4
	A2			8:488.0-503.0						
:	A3			8:503.0518.0						
	A4			8:518.0552.0				****		
1000	CS1 A2	76	38	10:306.0—315.0	100	1300	41 dB, H1	41 dB, H2	51 dB, H3	42 dB, H4
500	CS1 A1	76	38	10:020.0—044.0	100	1300	50 dB, H1	45 dB, H2	60 dB, H3	46 dB, H4
	CS1 A2			10:044.0057.0						,
550	CS1 A1	76	38	10: 57.0-74.0	100	1300	50 dB, H1	45 dB, H2	60 dB, H3	46 dB, H4
	CS1 A2			10:057.0—074.0						
600	CS1 A1	76	38	10: 74.0—90.0	100	1300	50 dB, H1	45 dB, H2	60 dB, H3	46 dB, H4
	CS1 A2			10:105.0—116.0						
650	CS1 A1	76	38	10:116.0—130.0	100	1300	. 41 dB, H1	41 dB, H2	51 dB, H3	42 dB, H4
	CS1 A2			10:130.0—145.0						
Hz	#	#	cm/s		Hz	Hz	Gain, Receiver			

## Appendix C

# Seneca-3 test schedule and recording parameters

Table C.1: Seneca–3 Data Runs and Recording Parameters.

Dat	a Run		Tape Desc	Fil	ters		Recording Chan	nel Description	
•	Cloud	Spd	Count	нр	LP	Ch 1	Ch 2	Ch 3	Ch 4
1 0,	A1	- Spa	7:156.0—165.0			-			
	A2		7:165.0—171.0			20 dB, H1	46 dB, H1	20 dB, H2	46 dB, H2
1500	A3	19	7:171.0-178.0	0	2500	мв	EV	мв	EV
	A4		7:178.0—184.0						
	<b>A</b> 1		7:204.8—214.0	,					
	A2		7:214.0—221.0	_	0500	20 dB, H1	46 dB, H1	20 dB, H2	46 dB, H2
1300	A3	19	7:221.0—232.0	0	2500	MB	EV	мв	EV .
	A4		7:232.0-239.0						
	<b>A</b> 1		7:239.0-248.0						
	A2		7:248.0-252.0		0500	20 dB, H1	43 dB, H1	20 dB, H2	46 dB, H2
1100	А3	19	7:257.0—263.0	O	2500	мв	EV	мв	EV
	A4		7:263.0—271.0						
	A1		7:271.0-279.0						
1000	A2	19	7:279.0-286.0	0	2500	20 dB, H1	43 dB, H1	20 dB, H2	46 dB, H2
1000	А3	19	7:286.0—294.0	ľ	2300	МВ	EV	МВ	EV
	A4		7:294.0—301.0					-	
	A1		7:301.0-310.0						
900	A2	19	7:310.0—317.0	0	2500	20 dB, H1	43 dB, H1	23 dB, H2	46 dB, H2
900	A3	13	7:317.0—325.0		2000	MB	EV	МВ	EV
	A4		7:325.—332.6						
	<b>A</b> 1		7:333.0341.0					,	
1200	A2	19	7:341.0-348.0	0	2500	20 dB, H1	43 dB, H1	20 dB, H2	46 dB, H2
1200	<b>A</b> 3	13	7:348.0-356.0		2000	MB	EV	МВ	EV
	A4		7:356.0—362.0						
	A1		7:362.0—371.0						
800	A2	19	7:371.0-378.0	0	2500	23 dB, H1	46 dB, H1	23 dB, H2	50 dB, H2
500	А3	"	7:378.0-385.2		2300	МВ	EV	МВ	EV
	A4		7:385.3-392.4						
	<b>A</b> 1		7:392.1-401.0						
700	A2	19	7:401.0-409.8	0	2500	23 dB, H1	50 dB, H1	23 dB, H2	53 dB, H2
'00	A3	1.0	7:409.8-417.8		-300	МВ	EV	МВ	EV
	A4		7:417.8—426.1						
	A1		7:426.1—435.7						
600	A2	19	7:435.7—442.8	О	2500	23 dB, H1	50 dB, H1	23 dB, H2	53 dB, H2
	A3		7:442.8—452.8			МВ	EV	МВ	EV
	A4		7:452.8-459.0	-					
Hz	#	cm/s		Hz	Hz	Gain, Receiver	<u> </u>	1	L

Continued on next page.

Table C.1: (continued)

Data	Run	,	Tape Desc	Fi	lters		Recording Chan	nel Description	
FQ (	Cloud	Spd	Count .	HP	LP	Ch 1	Ch 2	Ch 3	Ch 4
550	A1 A2 A3	19	7:459.0—468.1 7:468.0—476.1 7:476.1—484.6	0	2500	23 dB, H1 MB	50 dB, H1	23 dB, H2 MB	53 dB, H2 EV
	A4		7:484.6491.5					•	
	A1		7:491.5—499.5						
	A2		7:499.0—507.1		0700	26 dB, H1	50 dB, H1	26 dB, H2	53 dB, H2
500	A3	19	7:507.1515.6	0	2500	мв	EV	мв	EV
	A4		7:515.6—522.4						
450	A1 A2 A3	19	7:522.4—531.8 7:531.8—539.6 7:539.6—548.0	0	<b>2</b> 500	30 dB, H1 MB	50 dB, H1	30 dB, H2	53 dB, H2 EV
	A4		7:548.0—555.1						
400	A1 A2 A3 A4	19	7:555.1—565.0 7:565.0—573.1 7:573.1—580.0 7:580.0—587.4	ο	2500	30 dB, H1 MB	50 dB, H1 EV	30 dB, H2 MB	53 dB, H2 EV
350	A1 A2 A3 A4	19	7:593.8—602.8 7:602.8—609.2 7:609.2—616.7 7:616.7—623.0	0	2500	30 dB, H1 MB	50 dB, H1	30 dB, H2 MB	53 dB, H2 EV
300	A1 A2 A3 A4	19	7:623.0—631.7 7:631.7—639.8 7:639.8—647.2 7:647.2—654.9	ō	2500	30 dB, H1 MB	50 dB, H1 EV	30 dB, H2 MB	53 dB, H2 EV
250	A1 A2 A3 A4	19	7:654.9—664.0 7:664.0—673.7 7:673.7—680.7 7:680.7—688.0	ō	2500	30 dB, H1 MB	50 dB, H1 EV	30 dB, H2 MB	53 dB, H2 EV
1400	A1 A2 A3 A4	19	8: 10.0— 19.2 8:019.2—027.2 8:027.2—035.5 8:035.5—042.6	O	2500	20 dB, H1 MB	43 dB, H1 EV	20 dB, H2 MB	43 dB, H2 EV
Hz	#	cm/s		Hz	Hz	Gain, Receiver			

Continued on next page.

Table C.1: (continued)

Dat	a Run	-	Tape Desc	Fil	ters		Recording Char	nnel Description	
FQ	Cloud	Spd	Count	нр	LP	Ch 1	Ch 2	Ch 3	Ch 4 .
3000	A1 A2 A3 A4	19	8:042.6—050.9 8:050.9—057.8 8:057.8—064.8 8:064.8—071.4	0 4000		20 dB, H1	43 dB, H1 EV	20 dB, H2 MB	43 dB, H2 EV
3500	A1 A2 A3 A4	19	8:071.4079.1 8:09.1086.2 8:086.2093.5 8:093.5101.0	.0	5000	20 dB, H1 MB	43 dB, H1 EV	20 dB, H2 MB	43 dB, H2 EV
2500	A1 A2 A3 A4	19	8:101.0—108.4 8:108.4—115.7 8:115.7—123.4 8:123.4—129.9	0	4000	20 dB, H1 MB	43 dB, H1 EV	20 dB, H2 . MB	43 dB, H2 EV
2000	A1 A2 A3 A4	19	8:129.9—140.2 8:140.2—147.0 8:147.0—155.0 8:155.0—161.7	0	3500	20 dB, H1 MB	43 dB, H1 EV	20 dB, H2 MB	43 dB, H2 EV
Hz	#	cm/s		Hz	Hz	Gain, Receiver			

### Appendix D

## Error analysis methods

A simple and straightforward error—analysis was performed in order to quantify the accuracy and precision of the measured data and calculated results. A brief description of the fundamental measurements performed, quantifiable errors in these measurements, and the propagation of these errors is described in this section.

### D.1 Directly measured quantities

Several direct measurements were performed during the course of this experiment, including, but not limited to: the mass volume of air injected into each bubble cloud released into Lake Seneca, the bubble radius distribution released from a single needle tip, the radius—height profile of a rising bubble cloud, and the incident main bang and backscattered voltages measured from the receiver outputs. In such cases where the number of measurements performed was large (such as the measurement of the total gas volume and of the bubble radius dis-

tribution), the accepted experimental value and uncertainty were determined statistically from the mean and standard deviation of the measurements. For example, the measurement of the total gas volume of a single bubble cloud was described in Chapter 4.3.6. In this case, a large number of measurements were performed with a precisely calibrated instrument (the electronic mass flow meter). Over the course of the experiment, the mean total gas volume injected into the bubble cloud is  $\overline{V_{\rm gas}} = 1.22\,l$ , and the standard deviation of these measurements is  $dV_{\rm gas} = .014\,l$ . Therefore, the accepted experimental value for the total gas volume is  $1.22\pm.014\,l$ .

In other cases where the sample size was relatively small (< 10 samples), the experimental error was estimated from some systematic uncertainty (*i.e.*, limited by the resolution of the measurement device) or, where appropriate, a "best–guess" error estimate (*e.g.*, the radial profile of the bubble cloud is not clearly defined, as described in Chapter 4.3.4). In this case, the error introduced by the non–distinct cloud boundary ( $\approx \pm 1c_e$ ) exceeded the systematic error of the measuring device (a mm scaled ruler).

#### D.2 Derived quantities

In addition to the basic measurements described above, there are a host of derived quantities which are dependent on the directly measured values. These include, but are not limited to, derived quantities such as the mixture sound speed, the volume of the scatterer, the resonance frequency of the bubble cloud, the target strength, etc. All of these quantities are derived from the

equations specified in the text. The uncertainties in the basic measurements are propagated through the equations using a Taylor series expansion to first order in the perturbed quantities. For example, the uncertainty in the void fraction is determined by the combined uncertainty in the total gas volume measurement (directly measured), and the uncertainty in the net cloud volume (a derived quantity) as follows:

$$\beta \pm d\beta = \frac{V_{\text{gas}}}{V_{\text{cloud}}} \pm \left| \frac{V_{\text{gas}}}{V_{\text{cloud}}} dV_{\text{gas}} \right| \pm \left| \frac{V_{\text{gas}}}{V_{\text{cloud}}^2} dV_{\text{cloud}} \right|. \tag{D.1}$$

Here, the quantities preceded by d denote the deviation or uncertainty of the (directly or indirectly) measured values. The absolute value of each term was taken to effectively yield an upper bound on net error. The mean and uncertainty in the void fraction are calculated from

$$\beta = \frac{V_{\text{gas}}}{V_{\text{cloud}}} \tag{D.2}$$

$$\beta = \frac{V_{\text{gas}}}{V_{\text{cloud}}}$$

$$d\beta = \left| \frac{V_{\text{gas}}}{V_{\text{cloud}}} dV_{\text{gas}} \right| + \left| \frac{V_{\text{gas}}}{V_{\text{cloud}}^2} dV_{\text{cloud}} \right|.$$
(D.2)

This technique for propagating the experimental error was carried out for all derived quantities described in the text in a manner analogous to the one described above.

#### **Experimental Uncertainties** D.3

For the most part, the measurements processes are described in Chapter 4. The accepted experimental values and projected errors obtained from these measurements using the error propagation technique described above are summarized in Table 4.5. Listed below is a summary of how the uncertainties (errors) assigned to the various non-derived quantities, measured and otherwise, were determined.

- $V_{\rm gas}({\rm Gas\ volume})$  Chapter 4.3.6: The error in the total gas volume measurement was taken from the standard deviation of of 112 independent measurements as described in Chapter 4.3.6
- L (Cloud length) Chapter 4.3.3: The error in the cloud length was based
  on the error in the cloud velocity measurements and the error in the
  accuracy of the measured height of the camera deployed above the bubble
  maker.
- Bubble Size distribution Chapter 4.3.7: The error in the bubble size distribution measurements was determined from the standard deviation of the measured values.
- f (Acoustic driving frequency): The center frequency was determined from the readout of the "up/down" shifter for the NUWC parametric source. No uncertainty was assigned to it's measured value.
- MB (Main bang voltage) Chapter 5: For a given driving frequency
  the mean main bang voltage was determined by multiple measurements
  taken for repeated cloud releases as described. The uncertainty represents the spread in the maximum and minimum measured values observed at a given frequency.

- EV (Echo voltage) Chapter 5: For a given driving frequency the mean echo voltage was determined by multiple measurements taken for repeated cloud releases as described. The uncertainty represents the spread in the maximum and minimum measured values observed at a given frequency.
- d (depth) Chapters 4 and 5: The deployment depths for the hydrophones, sources, and the bubble maker was determined using a cable meter. The possible error in this measurement was not quantified.
- Po (hydrostatic pressure): The hydrostatic pressure was determined from the deployment depth. The possible error in this measurement was not quantified.
- c (liquid sound speed): The liquid sound speed versus depth profile was provided by NUSC daily. The possible error in this measurement was not quantified.
- $c_{\rm g}$  (sound speed of air): The ambient pressure dependent gaseous sound speed was determined from the well known relationships provided in texts (Kinsler *et al.*, 1982). No uncertainty was specified.
- $\rho$  (liquid density): The hydrostatic pressure dependent liquid density was determined from the well known relationships provided in texts (Kinsler et al., 1982). No uncertainty was specified.
- $\rho_{\rm g}$  (density of air): The hydrostatic pressure dependent density of air was determined from the well known relationships provided in texts (Kinsler

et al., 1982). No uncertainty was specified.

•  $\kappa$  (polytropic exponent): The polytropic exponent for air was calculated from equations specified in the text. No uncertainty was specified.

### Appendix E

## Rise-time dependent target strength measurements

Shown here are the time-elapsed backscattering data runs used to compose Figure 5.16. The target strength measurements were conducted during the Seneca-3 experiment using receiver H1, the closest to the target. The average target strength for these three runs is indicated by by the dashed line. It is clear that there is a significant deviation in the measured TS in the first few seconds after cloud release and again after 20 s. This variability is likely due to the low signal to noise present when the cloud is just entering and exiting the beam, as well as non-uniformities in the shape or localized distribution of the bubbles within the three clouds, as well as the possibility that the clouds did not rise along the same path. However, in bins 5–15, the measured TS for the three clouds is highly repeatable.

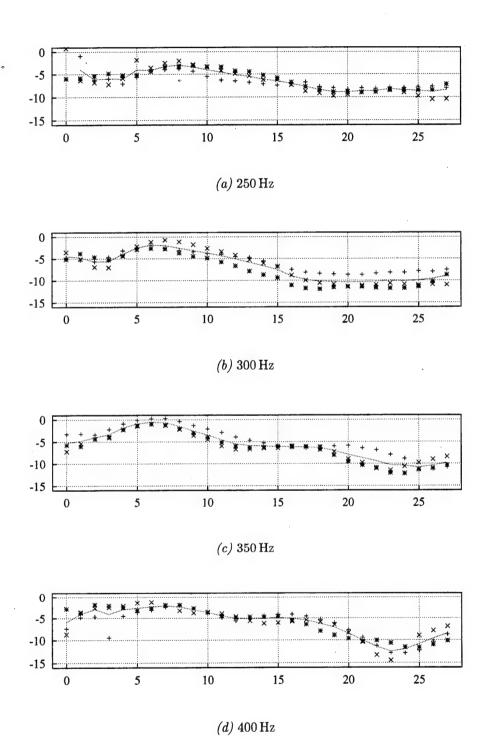
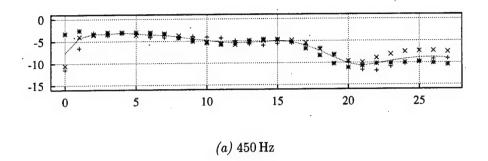
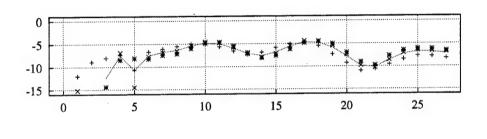
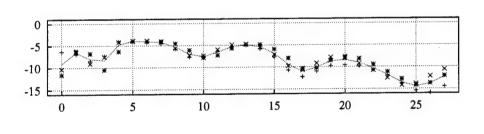


Figure E.1: Time elapsed TS (dB) versus rise time (s).





(b) 500 Hz



(c) 550 Hz

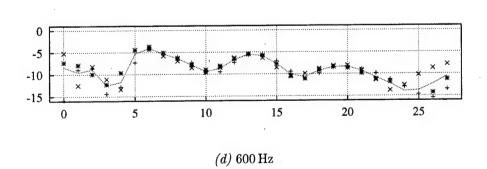
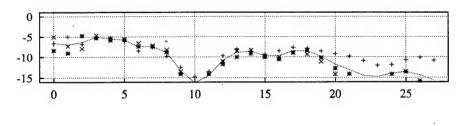
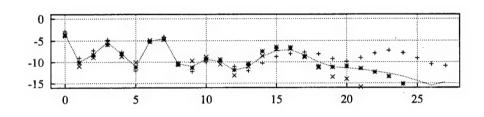


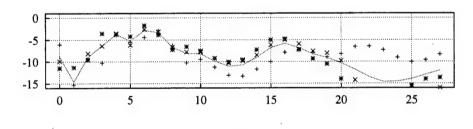
Figure E.2: Time elapsed TS (dB) versus rise time (s).



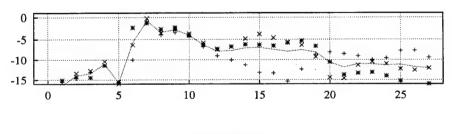
(a) 700 Hz



(b) 800 Hz

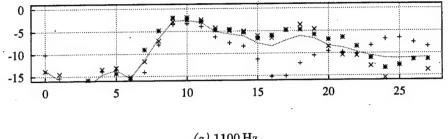


(c) 900 Hz

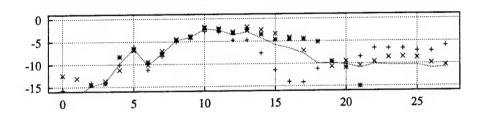


 $(d)~1000\,\mathrm{Hz}$ 

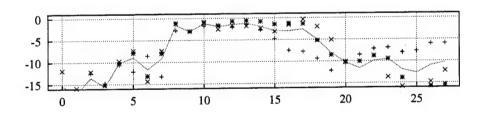
Figure E.3: Time elapsed TS (dB) versus rise time (s).



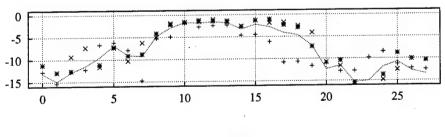
(a) 1100 Hz



(b) 1200 Hz

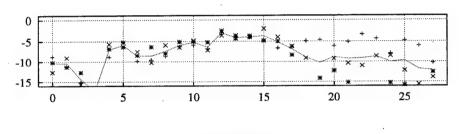


(c) 1300 Hz

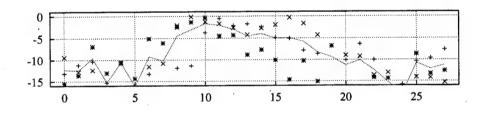


(d) 1400 Hz

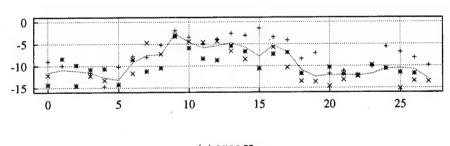
Figure E.4: Time elapsed TS (dB) versus rise time (s).



(a) 1500 Hz



(b) 2000 Hz



(c) 2500 Hz

Figure E.5: Time elapsed TS (dB) versus rise time (s).

## Appendix F

Source Codes

return(M);		***************************************	11. Jan 1997	SIT TADE THE TADE III)	(00 Table 10	14 (0>0.)	: DI-DI	else	U=0.1	· (11)			*D" nrestra	ELT TYPE But 20(ELT TYPE 20)	}	zo = 20;		return(20);	•		***************************************		extern "C"	FLI_TYPE puttheta(FLI_TYPE THEIA)	A L. C. C. POSTONIA .	return(theta);				extern "C"	FLT_TYPE putmu(FLT_TYPE MU)		return(mu);	,		n de la companya de l	extern "C"	tri iire puepaiteri iire rai)	obi-PHI:return(phi);		
xetum (ambur);		/	extern "C" rise worm 1.16.63	FLLLIFF Durcat(FLLLIFF delter)		return(df):			/+			putminmant(FLI_TYPE minf, FLI_TYPE maxf)	(d (min) < mark)	1		PHAX = BAX2:			alse	<b>~</b>	farm = maxf;	fMAX-minf;		return(0.);		÷.	***************************************		extern "C"	FLT_TYPE puta(FLT_TYPE A)		refull(a);				***************************************		THE TODE SHOW TWO ME	And the presentant and the second		ī
nscsp3.1.cxx	// ieff Schindall last rewiston 2DEC93	"	// ***********************************	***************************************	// v3.1 Now uses a separate file for phase speed calcs			// gausian or discrete bubble dist.	// w3.0 Now commolles under Symmantec C++ and gnuC++		// w2.2 Cleaned up program and Fixed Errors	// Can now run with or w/o bubble damping // compile on anchor, run on haral-or UV system.	// w2.1 included bubble damping Commander/Prosperetti	// and converted to C++ code.		// wi.0 First version for Mathilinki.2 c-source code.	"		#include "mscsp3.1.h"	#include 'nacepglob.b"	ELT TYPE E:		**************************************	chare IMPO MCSP()	{return "\n\nWSCSP 3.102 DEC 1993 jeff schindall\n\n";};		/* ***********************************	extern "C"	Chort anticacapicoratage tar_tire coratage.	- April 10 may - and -		if(cowerage==0)	}	WhitecapCoverageSwitch * O;	Toeston without the ore of		J	WhitecapCoverageSeitch = 1;	answer "Whitecap Coverage ON";	);	E

```
// Include adjustable z==0 plane so that I can look at the bubbler
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 /* this subroutine shall return the maximum IS below the monopole resonance frequency as given by the Minneart Expression.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           // Calculate Cm at 256pts(or 128 pts) btwn fMIN and fMAX
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               // Extrapolate Cm at each fq based on array of Cm(1).
                                                            void TSatResfq(INT_IYPE method, FLT_IYPE ans[2])
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               // Suggested changes to make program quicker:
                                                                                                                                                                                                            FLT_TYPE oldmu, oldphi,tmpTS;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             FLT_TYPE resfq(INT_TYPE method)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      FLI_TYPE ans[2];
if (free == 0)
TSatResfq(method,ans);
                                                                                                                                               ISff(INT_IYPE method)
                                                                                                                                                                                                                                                                                                                      tmpTS=IS(method);
                                                                                                                                                                                                                                                                                                                                                           putmu(oldmu);
putphi(oldphi);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          return(ans[1]);
return(enswer);
                                                                                                                                                                                                                                                                                                                                                                                                     return(tmpIS);
                                                                                                                                                                                                                                                       oldphi-phi;
                                                                                                                                                                                                                                                                                                putph1(0.);
                                                                                                                                                                                                                                                                           putmu(0.);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       extern "C"
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      extern "C"
                                                                                                      exterm "C"
                                                                                                                         FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      case 4: // Calculate Roughness using Assymptotic TS
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 +SurfaceRoughness(k,mu,z,phi)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     +SurfaceRoughness(k,mu,z,phi)
                                                                                                                                                                                                                                                                                                                      +SurfaceRoughness(k,mu,z,phi)
                                                                                                                                                                                                                                                                                                                                                                                                                          +SurfaceRoughness(k,mu,z,ph1)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          answer" (TSanderson(k, h, g, a, theta, M)
                                                                                                                                                                                                                                                                                                                                                                                                     ansver=(TSanderson(k,h,g,a,theta,H)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       case 3: // Calculate Roughness
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 mu = mu_approx( U, kz );
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      mu - mu_approx(U, kz );
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            answer-(TSassym(k,b,g,a)
                                                                                                                                                                                                                                                                                                answer=(TSassym(k,h,g,a)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               kz = k*sin(phi);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  kz - kesin(phi);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           ansver--9999.;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            putmu(mu);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        putmu(mu);
                                                         g = rhom()/rhol();
                                                                                                                                                                                                              answer = -9999.99;
                                                                                                                                                                                                                                                                                                                                                           break;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                break;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  break;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 break;
                                                                                                                                                                                                                                     switch (method)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      default:
                                                                                                                                                                     h = cm(w)/cl;
                                                                                                                                                                                                                                                                             case 1:
                                                                                                                                                                                                                                                                                                                                                                                   cese 2:
                                                                                                      z=depth(Po);
                                                                                                                         v = 2*PI*f;
                                                                                                                                               r = v/cl:
complex
                                                                                                                                                                                                                                                                                                                                                         returnval = WhitecapCoverageSwitcheiO.
*logiO(3.66*.000001*exp(3.41*log(U) ));
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          return( getcm(f)/(2 * PI * a) * sqrt(3 * rhom()/rhol()));
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           return( (Press-1.e5)/(rhol() *9.81) );
                                                                                                                                                                                                                                                       FLT_TYPE WhitecapCoverageIndex()
                                                                                FLT_TYPE depth(FLT_TYPE Press)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     FLT_TYPE IS(INT_TYPE method)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 FLI_TYPE fo(FLI_TYPE f)
                                                                                                                                                                                                                                                                                                FLI_TYPE returneal;
                                                                                                                                                                                                                                                                                                                                                                                                                                              returnval=0;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         return(returnval);
                                                                                                                                                                                                                                                                                                                                          1f (UI=0.)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             extern "C"
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 extern "C"
                                                                                                                                                                                                                                                                                                                                                                                                                              •;;•
```

```
// Calculate Darivative of Bessel and Weumann function of order m_{\star} // argument ks_{\star} kah_{\star}
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     LPm = ((2.eu-1.)exrelPmm1-(m-1.)elPmm2)/m;
// Calculate Bessel and Neumann function of order m+1,
// argument ke, kah.
                                                                                                                                                                                                                                                                                                                                                                                                                                                // Calciulate Legendre Polynomial of order m, argument
// xx=com(Pl-theta)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        jpm=(me jmm1 - (m+1.) * jmp1)/m2p1;
npm=(me nmm1 - (m+1.) * nmp1)/m2p1 ;
jpem=(me jemm1 - (m+1.) * jemp1)/m2p1 ;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                c= (jpem_jpm_jemenm - npw/jpmegh)
/(jpem_jpm_jemejm - gh);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 jempi = (m2pi)/(keh)*jem-jemmi;
                                                                                                                                                                                                                                 // Basis for Legendre Polynomials
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       jmp1 = (m2p1)/(km)*jm-jum1;
nmp1 = (m2p1)/(km)*nm=nmmm1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            jpem_jpm_jem=jpem/jpm/jem;
                                                                                                                                                                                                                                                                                                                                                 for (m=0;m<=KUMBESS;++m)
                                                                                                                                                                                                                                                                                                                                                                                                            m2p1=2.0+m+1.0;
                                                                                                                                                                     jemi-cos(kah)/kah;
                                                                                                                                                                                                                                                                         xx=cos(PI-theta);
                                                                                                                                                                                            jem-sin(kah)/kah;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        switch (mkt)
                                                                                                                                  nme -cos(ka)/ka;
                                                      juni-cos(ka)/ka;
                                                                                                                  nmm1-sin(ka)/ka;
                                                                            jm-sin(ka)/ka;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          if (m>0)
              kah = ka/h;
                                                                                                                                                                                                                                                                                                             LPm = 1;
WUMBESS=M;
                                                                                                                                                                                                                                                                                             LPmm1=1;
                                                                                                                                                                                                                                    SCC(FLT_TYPE ka, FLT_TYPE g, complex h, FLT_TYPE theta, int M)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    // Basis for Spherical Bessel and Neumann functions
                                                                                                                                                                                              INT_TYPE m, NUMBESS;
                  fres = fq;
maxTS=tstmp;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    r=complex(0.,0.);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               jpen_jpm_jem,
                                                                            );
ans[0] = maxTS;
                                                                                                                  ans[1] = fres;
                                                                                                                                                                                                                                                                                                                   minusitom,
                                                                                                                                                                                                                                                                                             FLT_TYPE
                                                           ::
                                                                                                                                                                                                               FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                        fmin=(0.9*.25 *cl/(depth(Po)*sin(phi)));
                                                                                                                                                                                                                                                                                                                                 if (fMAX > frapres) maxf = 1.1sfmpres;
                                                                                                                                                                                                                                                                                                                                                                                                          if ((depth(Po)==0.)}|(phi==0.))
fmin = .8efmpres;
// as a possible reflecting plane.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  fq=putf(fq+df);
tstmp = TS(method);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             df = (maxf - fmin)/DF;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              if (tstup > maxTS)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    if (fmpres < fmin)
fmin = .8*fmpres;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         maxTS = TS(method);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               /* fmin -fMIN; */
                                                                                                                                                                                                                                                                           fq;
fmpres = fo(1.);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              fmine- fmin;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            while (fq<maxf)
                                                                                                                                                                                            maxI3,
tetmp,
maxf,
ansver[2],
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   fq-putf(fmin);
                                                                                                                                                                                                                                                                                                                                                         fres - fmpres;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              if (fmin <0)
                                                                                                                                                                                                                                                                                                                   mexf = fHAX;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           DF-1024;
                                                                               INT_TYPE
                                                                                                                                        FLT_TYPE
```

```
// Calculate Derivative of Sessel and Wemmann function of order m, // argument km, kmh.
                                                                                                                                                                                                              LPm = ((2.em-1.)*xxelPumi-(m-1.)*LPum2)/m;
// Calculate Bessel and Neumann function of order m+1,
                                                                                                                                          // Calciulate Legendre Polynomial of order m_{\star} argument // xx=cos(PI-theta)
                                                                                                                                                                                                                                                                                                                                                                                                                                                       c= (jpem_jpm_jemenm - npm/jpmegh)
/(jpem_jpm_jemenm - gh);
                                                                                                                                                                                                                                                                                                                                                               jemp1 = (m2p1)/(kab)*jem-jemm1;
                                                                                                                                                                                                                                                                                                              jmp1 * (m2p1)/(ka)*jm-jmm1;
nmp1 * (m2p1)/(ka)*nm-nmm1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             jpem_jpm_jem=jpem/jem;
                                                                                                                                                                                                                                                                              hmp1 = (m2p1)/(kr)*hm-hmm1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             minusitom-1.;
break;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        case 0:
minusitom=1.;
                                                                                                                                                                                                                                                 // argument km, kah.
                                                               for (m=0;m<=NUMBESS;++m)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      wuitch (m#3 )
{
                                                                                                                m2p1=2.0+m+1.0;
xx=cos(PI-theta);
LPmni=1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            switch (m&1)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            case 1:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           bresk;
                                                                                                                                                                                                 1f (m>0)
                               LPm = 1;
NUMBESS=M;
                                                                                                                                                                                                                                                                                                                                                                                                                               // Basis for Spherical Bessel and Meumann functions
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            hami = cos(kr)/kr + I * sin(kr)/kr;
hm = sin(kr)/kr - I * cos(kr)/kr;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            // Basis for Legendre Polynomials
                                                                                                                                 INT_TYPE = NUMBESS;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           jemm1=cos(kah)/kah;
jem=sin(kah)/kah;
                                                                                                                                                                                                                               gh,
jpem_jpm_jem,
keh,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            jmmi-cos(ka)/ka;
jm-sin(ka)/ka;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           nmisin(ka)/ka;
nm -cos(ka)/ka;
                                                                                                                                                                                                                                                                                                                                                                                                                recomplex(0.,0.);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              kah = ka/h;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             ka = k*a;
kr = k*R;
                                                                                                                                                                                                                                                                                                                                                                                                                                                               :9•q - q8
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             SCOM (FLI_ITPE k,FLI_ITPE a,FLI_ITPE R,FLI_ITPE g, complex h, FLI_ITPE theta, int M)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             case 1:
minusitom-1.;
                                                                                  minusitom-1.;
                                                                                                                                                                                                                                                                                                                                                                                                                };
ans = (abs(2er/ka));
if (ans <= 0.)</pre>
                                                                                                                                                                                                                                                                                                                                                                                                                                                            return( -9999.99);
                                                                                                                                                                                                                                                                                                                                                             LPmm2-LPmm1;
LPmm1- LPm;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             return( (ens) );
                                                                                                                                                                                                                                                                                               jarjapi;
narnapi;
jemrjeapi;
                                                                                                                                                                                                                                                                 jemi-jem;
                                                                                                                                                                                                                                 jami-ja;
nami-na;
                                                                  case 0:
                                                                                               bresk;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               minusitom,
                                                                                                                                                                                                 1f (N>0)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                FLT_TYPE
```

```
LPm = ((2.*m-1.)*xx*[Pami-(m-1.)*[Pam2)/m;
// Calculate Bessel and Neumann function of order m+1,
                                                                                                                                                      // Calclulate Legendre Polynomial of order m, argument
// xxwcos(PI-theta)
                                                                                                                                                                                                                                                                                                                                                                            // Calculate Derivative of Bessel and New
                                                                                                                                                                                                                                                                                                                                                                                                                           jpa=(me jmal - (m+1.) * jmpl)/m2p1;
npa=(me nmal - (m+1.) * nmpl)/m2p1;
jpa=(m* jemml - (m+1.) * jempl)/m2p1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            r += minusitom=m2pi/(1+I*c)*LPm;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             c= (jpem_jpm_jemenm - npm/jpmegh)
/(jpem_jpm_jemejm - gh);
                                                                                                                                                                                                                                                                                                                                              jemp1 = (m2p1)/(kah) \cdot jem-jemn1;
// Basis for Legendre Polynomials
                                                                                                                                                                                                                                                                                             jmp1 = (m2p1)/(km)*jm-jmm1;
nmp1 = (m2p1)/(km)*nm-nmm1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              jpem_jpm_jem=jpem/jpm/jem;
                                                                                                                                                                                                                                                                argument ke, ksh.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            if(c != complex(0.,1.))
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             minusitom-1.;
                                                                                                                                                                                                                                                                                                                                                                                           // argument ka, kah.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             case 0:
minusitom=1.;
                                                                                for(m=0;m<=NUMBESS;++m)
                                                                                                                                 m2p1=2.0em+1.0;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           switch (m&1)
                                xx=cos(PI-theta);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            case 1:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            break;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             breek;
                                                                                                                                                                                                               1f (m>0)
                                                LPmi-1;
                                                                 LPm = 1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           WOWBESS-M;
recomplax(0.,0.);
// Basis for Spharical Bessel and Neumann functions
                                                                                                                                                                                                                                                                                                                                                                                                                                                                            // woid ejm(IRT_IYPE, FLT_IYPE, *FLT_IYPE);
// woid sym(IRT_IYPE, FLT_IYPE, *FLT_IYPE);
                                                                                                                                                                                                                                                                                                                                               INT_TYPE m, mm1, mp1, NUMBESS;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         jemmi=cos(kah)/kah;
jem=sin(kah)/kah;
                                                                                                                                 jpe,

pp.

gh,

jpen,

jeen,

jeen,

jeen,

jeen,

sel-selli,

en-selli,

en-selli,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         nami=sin(ka)/ka;
na= -cos(ka)/ka;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        jumi-cos(ks)/ks;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           je-sin(ka)/ka;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             keh - ka/h;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             gh - h•g;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             SCCr(FLI_TYPE ka,FLI_TYPE g, FLI_TYPE b, FLI_TYPE theta, int M)
                                                                                                                                                                                                                              ans = (abs( 2er/km * kr ));
if (ans <= 0.)
return( -999.99);
                                                                                                                              break;
case 3:
I2me-I;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                            LPmm2-LPmm1;
LPmm1- LPm;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            return( (ans) );
case 0:
I2m=1.;
                                                                                                               12m--1.;
                                                                                                                                                                                                                                                                                                                                                                                                                                narnapi;
jenrjempi;
                                                                                                                                                                                                                                                                                                                                                                jemmi=jem;
                                                                 12m-1;
                                                                                                                                                                                                                                                                                                                              jmmi-jm;
nami-nam;
                                                case 1:
                                break;
                                                                                                 case 2:
                                                                                                                                                                                                                                                                                                                 ham1-ha;
                                                                                                                                                                               bresk;
                                                                                break;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               minusitom,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            ans,
xx,
m2p1,
LPm,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             .1:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              FLT_TYPE
```

```
// Calculate Derivative of Bessel and Neumann function of order m,
                                                                                                                                                                                                                                                                                                                    | jpam=(m* aj[mni] - (mpi) * aj[mpi])/m2pi;
| npum=(m* an[mni] - (mpi) * an[mpi])/m2pi;
| jpam=(m* aj*[mni] - (mpi) * aj*[mpi])/m2pi;
                                                                                                                                                                 |pem=( - (mpl) * sj(mpl])/m2pl;
| npm=( - (mpl) * sn(mpl])/m2pl;
|pem=( - (mpl) * sje(mpl])/m2pl;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  if(c != complex(0.,1.))
r = minusitomemZpl/(i-lec)elPm;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          ce (jpem_jpm_jemena - npm/jpmegh)
/(jpem_jpm_jemesj[m] - gh);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           minusitom= (((m&1)==0) 7 1:-1 );
                                                                                                                                                                                                                                                                                                                                                                                                                                                 jpem_jpm_jem=jpem/jpm/sje[m];
                           // argument ke, keb.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            LPmm2=LPmm1;
LPmm1= LPm;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          jurjupi;
nurnupi;
jeurjeupi;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   jemm1=jem;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  };
/* fre(sn);
free(sj);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               1f (N>0)
                                                                                                                              1f (n==0)
                                                                  met-m-1;
                                                                                     12+m-1dm
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            // Calciulate Legendre Polynomial of order m, argument
                                                                                                                                                                                                                                                                               // Basis for Spherical Bessel and Meumann functions
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                LPm = ((2.4m{-}1.)*x*LPmn1{-}(m{-}1.)*LPmn2)/m;
                                                                                                                                                                       // void sju(INT_TYPE, FLT_TYPE, *FLT_TYPE);
// void syn(INT_TYPE, FLT_TYPE, *FLT_TYPE);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       jemp1 = (m2p1)/(keh)*jem-jemm1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               // Basis for Legendre Polynomials
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       japi = (m2pi)/(ka)*ja-jami;
napi = (m2pi)/(ka)*na-nami;
   INT_TYPE m, mmi, mp1, NUMBESS;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 // xx=cos(PI-theta)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   m2p1=2.0em+1.0;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  jemmi=cos(kah)/kah;
jem=sin(kah)/kah;
                                                                                                                                                                                                                                                         r=complex(0.,0.);
                                                                                                                                                                                                                                                                                                                                                                                                        jumi"cos(ka)/ka;
ju-sin(ka)/ka;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                    nmmi-sin(ka)/ka;
nm--cos(ka)/ka;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          xx=cos(PI-theta);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        for(m=0;m<=H;++m)
                                                                                                                                                                                                                                                                                                                                                                 kah - ka/h;
                                                                                                                                                                                                                                      NUMBESS-N;
                                                                                                                                                                                                                                                                                                                          gh - h•g;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               LPmmi-1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   LPm - 1;
                                                                                                                                                                                                                                                                                                                                                                                                          SCCrmodes(FLI_TYPE ka, FLI_TYPE g, FLI_TYPE b, FLI_TYPE theta, int M)
                                                                                                                                                                                                                                                                                                  return( -9999.99);
                                                                                                                                                                       LPmm2=LPmm1;
LPmm1= LPm;
                                                                                                                                                                                                                                                       ans = (abs(2*r/ks));
                                                                                                                                                                                                                                                                                                                                           return( (ans) );
                                                                                   jm=jmp1;
nm=nmp1;
jem=jemp1;
jmm1=jm;
nmm1=nm;
jemm1=jem;
                                                                                                                                                                                                                                                                               if (ans <= 0.)
                                                                                                                                                                                                                                                                                                                                                                                      FLT_TYPE
```

```
case 4: // Calculate Roughness using Assymptotic TS
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         +SurfaceRoughness(k,mu,z,phi)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    +SurfaceRoughness(k,mu,z,phi)
                                                                                                                                                                                                                                                   CalcClCm(); //calculates clcmlist if need be...
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               answer=(TSanderson(k,h,g,k,theta,H)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             ansver"(TSanderson(k,h,g,a,theta,N)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         +SurfaceRoughness(k, mu, z, phi)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               case 3: // Calculate Roughness
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   mu = mu_approx( U, kz );
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   answer-(TSassym(k,b,g,a)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                        h = 1/clcmlist[jj]; //cm(w)/cl;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            kz - kesin(phi);
                                                                                                                                                                                                                                                                                                                                                                                      for (jj=0;jj<NUMFQS;jj++) {
                                                                                                                                                                                                                                                                                                                                                                                                                                  w = 2*PI*fqlist[jj];
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          putmu(mu);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           Amswer = -9999.99;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       break;
                                                                                                                                                                                                                                                                                               g = rhom()/rhol();
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                switch(method)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        break;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              CAS. 1:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            case 2:
                                                                                                                                                                                                                                                                                                                                                                                                                                                           k = w/cl;
                                                                                                                                                                                                                                                                                                                                           z-depth(Po);
z,
kz,
snsver,
hi,
                                                                                                               complex
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   void TSFQlist(INT_TYPE method, FLT_TYPE efraqs, FLT_TYPE eTSlist, long num)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            top = \{a*gasgalakak/3*(1-z)/(1-z*a*gaskak/3-z*a*gasgalakakk/3)\};
                       ISnf(FLI_IYPE k, complex h, FLI_IYPE g, FLI_IYPE a, FLI_IYPE theta,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      TSassym(FLT_TYPE k.complex h.FLT_TYPE g. FLT_TYPE a)

// NOTE: Assymptotic expansion of m=0 term in TSanderson.

// See Moree and Ingard or Rehewkin for details.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        ans = SCCnf( k,a,r,g,h,theta,M)/2en;
if (ans <= 0.)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          return(10.*log10(norm(tmp)));
                                                                                                                                                                                                                                                                                                                                                                return( 20*log10(ans) );
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     //TSlist assumed predefined...
                                                                                                                                                                                                                                                                                                                      return( -9999.99);
                                            int M, FLT_TYPE r)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    FLT_TYPE dummy;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   complex z,tmp;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             int ind, jj;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           z=1/(g*b*b);
                                                                                                                                                                                                        ka = kea;
                                                                                                                                                                                                                             kr = ker;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            FLT_TYPE
                                                                                                                                       ka kr.
                                                                                                             FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    extern "C"
                                                                                                                                                                                                                                                                                                                                             .18
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   JSanderson(FLT_IYPE k,complex h, FLT_IYPE g, FLT_IYPE a,FLT_IYPE theta, int M)
                                                                                                                                                                                                                         FII_TYPE SurfaceRoughness(FLI_TYPE k.FLI_TYPE mu,FLI_TYPE z,FLI_TYPE phl) //
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        -4. * mu * sinkdphi*sinkdphi
                                                                                                                                                                                                                                                                          FLT_TYPE rghness, sinkdphi;
sinkdphi=sin( k * (z-z0) * sin( pbi));
                                                                                                                                                                                                                                                                                                                                                                rghness* (1.+ mu * mu + 2.* mu
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              ans = SCC( ke,g,h,theta,K)/2*a;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           return( 10*log10(rghness));
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      return( 20*log10(ans) );
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        return( -9999.99);
                                                                                                               return( -9999.99);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                return( -9999.9);
                                                                   ans = (abs(2*r/ka));
                                                                                                                                                            return( (ans) );
                                                                                                                                                                                                                                                                                                                                                                                                                                      rghness ** rghness;
                                                                                                                                                                                                                                                                                                                                                                                                                                                             if (rghness=0)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     if (ans <= 0.)
                                                                                          if (ens <= 0.)
free(sje);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 ka = kea;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       ans;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          *1**
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  FLT_TYPE
```

```
return "\n\nInitialzed for Lake Seneca Calcs\n\n";
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  return "\n\nInitialzed for Marine Calcs\n\n";
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            putBetalist(mybetas,6);
                                                                                 putBetaList(mybetas,6);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           mybetas[0]=0.0055;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   mybetas[2]=0.0011;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 mybetas[1]=1.00;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        mybetas[3]-0.00;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              mybetas[4]=0.00;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      mybetas [5]=0.00;
                                                                                                                                                                                                                                       cher * init_marine()
mybetas[3]=0.00;
                     mybetas[4]=0.00;
                                           mybetas[5]=0.00;
                                                                                                                                                                                                                                                                                                    mybetas[6];
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     putrhol(1024);
                                                                                                                           puttheta(PI);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   puttheta(PI);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                          putc1(1500.);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             putrhog(1.);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        void initNSCSP()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         puta(0.25);
                                                                                                                                                                                                                                                                                                                                                                                    put20(0.);
                                                                                                                                                                                                                                                                                                                                                                                                                                  putmu(0.);
                                                                                                                                                                                                                                                                                                                                            put2(0.);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     FLT_TYPE
                                                                                                                                                                                                                                                                                FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  extern "C"
                                                                                                                                                                                                                 extern "C"
/* replace each frequency with its Scattering Gross Section */ //printf('thetaO %f\te %f\th %f\th %d\n',thetaO, g,h,MM);
                                                                                                                                                                                                                                       void SCCKAmodelist(FLI_TYPE *KAs, long num, FLI_TYPE g,FLI_TYPE h,
                                                                                                                                                                                                                                                                                                                                              /* replace each frequency with its Scattering Gross Section */
                                                                                                                                                                                                                                                                                                                                                                    //printf('thetaO Xf\tg Xf\th Xf\tM Xd\n',thetaO, g,h,MM);
                                                                                                                                                                                                                                                                                                                                                                                                                                  dummy= SCormodes(KAs[ind],g,h,theta0,MM);
KAs[ind]=dummy;
                                                                                                                                                                                                                                                              FLT_TYPE theta0, long MM)
                                                                                   dumny= SCCr(KAs,g,h,theta0[ind],HM);
                                                                                                                                                                                                                                                                                                                                                                                          for (ind=0;ind<num;ind++)
                                           for (ind=0;ind<num;ind++)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              mybetas[0]=0.0054;
mybetas[1]=1.00;
mybetas[2]=0.0016;
                                                                                                           theta0[ind]-dummy;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 char * init_seneca()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            putcl(1421.5);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  putrhol(1024);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             mybetas[6];
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            putrhog(1.);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      puta(0.376);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       put20(87.6);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         put2(87.6);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                putmu(0.);
                                                                                                                                                                                                                                                                                                        int ind;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           extern "C"
                                                                                                                                                                                                                 extern "C"
                                                                                                                               .
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        wold SCothetalist(FLI_TYPE KAs, FLI_TYPE g.FLI_TYPE h, FLI_TYPE *theta0,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               /* replace each frequency with its Scattering Gross Section */ //printf('theta0 Xf\tg Xf\th Xf\th Xf\th Xf\th Xheta0, g,h,HH);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         wold SCCKAlist(FLT_TYPE *KAs, long num, FLT_TYPE g,FLT_TYPE h,
                                                                                 ansver~(TSassym(k,h,g,a)
+SurfaceRoughness(k,mu,z,phi)
                                                                                                                                                                                                                   case 5: // Calculate Wear Field TS
                                                                                                                                                                                                                                         answer=(TSnf(k,h,g,a,theta,M,r)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      dummy= SCCr(KAs[ind],g,b,thetaO,NN);
                                       - mu_approx( U, kz );
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             FLI_TYPE theta0, long MM)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                long num, long MM)
                     - kesin(phi);
                                                                                                                                                                                                                                                                                                                                              answer=-9999.;
break;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              for (ind=0;ind<num;ind++)
                                                                                                                                                                                                                                                                                                                                                                                                              );
TSlist[jj]=ensver;
                                                                                                                                                                                                                                                              ::
                                                                 putmu(mu);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            KAs [ind] -dummy;
                                                                                                                                                                                                                                                                                  break;
                                                                                                                                                                           break;
                                                                                                                                                                                                                                                                                                                            default:
                       £ £
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 FLT_TYPE dummy;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               FLT_TYPE dummy;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         int ind;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           int ind;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    extern "C"
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       extern "C"
```

	XXX,		xxx=20.* 1	xxx=20.* logio( SCCr(14.,1.,0.8,PI,0)/2.*0.376);	
RAIGO-1;   RAIGO-1;   RAIGO-1;   RAIGO-1;   RAIGO-30;   RAIGO-30		eriables	xxx=14./0. xxx=Senecs	376; IS(xxx,0.376,1.,zzz,89.,0);	INT_TYPE
### STORINGE TO SET THE PART OF THE PROPERTY O	char* ccc;		KAs[0]-1.; KAs[1]-2.; KAs[2]-3.;		eSpCoverageSe
Intransect();	INT_TYPE		num=3; SCCKAlist(R putmu(0.);	0.5, 0.5,	
	111, num;		init_senece	9;	Seneca Corrections.cxx
ISCSPMain	сощравж		KAs [0]=300;		Sinclude "nacep3.1.h" Sinclude "nacep2lob.h"
by datault turn off Windspeed   IISCSPMain3.1.c	222;		// TSFQlist(5,	KAB,KAB,50)	
/ By default turn off Windspeed inSCSPMain3.1.C	r = 10.19;		_		TYPE TO THE SET OF THE PARTY AND A SET OF THE SET OF THE PARTY AND THE SET OF THE SE
ppMain3.1.c  ipglob.cxx  "ascep3.1.r"  "uscep5!ob.b"  // Calculated resonance fq of spherical cloud  // Wesumbar in host fluid 2 Pi fq/Cl  // Surface reflection coefficient -i< m < 0  // Surface reflection coefficient -i< m < 0  // Surface reflection coefficient -i< m < 0  // Surface reflection coefficient peak in TS(fq)  // Change in fq used to calculate peak in TS(fq)  // Change in fq used to calculate peak in TS(fq)  // Set zero plans	xxx=1.E30;				Senecaloviti. Into Kartilita estrullita grouppea priilita lejanilita
By default turn off the whitecep coverage	0				{ FLI_TYPE
// Set Density of host liquid // Set Density of host liquid // Set Density of host liquid // Rubber of Iterations in Solution INSCSPBIOD.CXX // Scattering Direction Pi-SachGeatter // Surface Crazing Angle // Surface Crazing Angle // Surface Crazing Angle // Surface Tellaction coefficient // Set the dalta frequency Tange // Surface Tellaction coefficient -1c mm < 0 phi, // Surface Tellaction coefficient -1c mm < 0 phi, // Surface Crazing angle an, // Surface Crazing angle // Change in queed to calculate peak in TS(qq) // Change in queed to calculate peak in TS(qq) // Set Tellaction coefficient -1c mm < 0 phi, // Set Tellaction coefficient -1c mm < 0 phi, // Set Tellaction coefficient -1c mm < 0 phi, // Surface Crazing angle // Change in queed to calculate peak in TS(qq) // Set Tellaction coefficient -1c mm < 0 phi, // Set Tellaction calculate peak in TS(qq)	.; tecapCoverageSwitch=0;	// By default turn off Windspeed // turn off the whitecap coverage	nscspl	Main3.1.c	is
// Set Density of gas in nucose // Set Density of bost liquid // Rubbe of Iterations in Solution INSCSPBIOD.CXX // Scattering Direction Pi-Bacificates // Surface Crazing Angle // Surface Crazing Angle // Surface Telection coefficient // Surface Telection coefficie	•				Yr,
// Rumber of Iterations in Solution TISCSPBIOb.cxx // Scattering Direction Pi-BackScatter // Surface Totaling Direction codificient // Surface reliaction codificient // Surface reliaction codificient // Surface reliaction codificient // Set initial iterquency // Set the initial iterquency range // Set the initial iterquency // Set the initial iterquency // Set the dalta iterquency // Set the initial iterq	<pre>= putrhol(1000.);</pre>	// Set Density of Most liquid			Y1,
//glob.cxx // Scattering Direction Pi-SackGatter // Surface Grazing Angle // Surface Crazing Angle // Surface Tellection coefficient // Surface Tellection coefficient // set the initial frequency FELTTYPE // Max and Minimum frequency Tange free, // Calculated resonance fq of spharital cloud // set the dalta frequency Tange free, // Scattering angle a, // Scattering angle a, // Stattering angle and, // Surface Tellection coefficient -1c mm < 0 phi, // Surface Tellection calculate peak in TS(fq) fMIX, // Minimum fq used to calculate peak in TS(fq) fMIX, // Stattering angle to calculate peak in TS(fq) fMIX, // Stattering angle to calculate peak in TS(fq) fMIX, // Stattering angle to calculate peak in TS(fq) fMIX, // Stattering angle to calculate peak in TS(fq) fMIX, // Stattering plane	- putM(20);	// Number of Iterations in Solution	nscspe	dob.exx	Ya. Xa.
// Surface Grazing Direction Pi-BacidCatter // Surface Crazing Angle // Surface Tellaction coefficient // Surface Tellaction coefficient // set the initial frequency Fauge // set the dalta frequency Tange // Wavenumber in host fluid 2 Pi fq/Cl // set the dalta frequency Tange // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Gloulated resonance fq of sphartcal cloud // Wavenumber in host fluid 2 Pi fq/Cl theta, // Wavenumber in host fluid 2 Pi fq/Cl theta, // Wavenumber in host fluid 2 Pi fq/Cl theta, // Wavenum 2 Pi fq/Cl theta, // Wavenum 2 Pi fq/Cl theta, // Wavenum 2 Pi fq/Cl theta, // Wavenumber 2 Pi fq/Cl theta, // Wavenum 2 Pi fq/Cl theta, // Wavenumber 2 Pi fq/Cl theta, //			//glob.cxx		ř.,
// Surface Grazing Angle // Surface Tellection coefficient // set the initial cloud radius // set the initial frequency range free, // Calculated resonance fq of spharital cloud // set the dalta frequency range free, // Settering angle a, // Settering angle mu, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection calculate peak in TS(fq) ff. // Change in fq used to calculate peak in TS(fq) ff. // Set zero plane zo, // Set zero plane	<ul><li>puttheta(PI);</li></ul>	// Scattering Direction Pi-BackScatt			i Kr.
// Surface reflection coefficient // set the initial frequency range from // Calculated resonance fq of spharital cloud // set the dalta frequency range free, // Calculated resonance fq of spharital cloud // set the dalta frequency range free, // Settering angle a, // Settering angle a, // Surface reflection coefficient -1c mm < 0 phi, // Surface reflection coefficient -1c mm < 0 phi, // Surface reflection calculate peak in TS(fq) ffMX, // Maximum fq used to calculate peak in TS(fq) df, // Change in fq used to calculate peak in TS(fq) df, // Set zero plane	- putphi(0.);	// Surface Grazing Angle	#include "nace		A1. Xb.
// set the initial frequency FET_TYPE // set the dalta frequency range free, // Calculated resonance fq of spharital cloud // set the dalta frequency range free, // Settering angle a, // Settering angle a, // Settering angle an, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface reflection coefficient -1< mm < 0 phi, // Surface freely and to calculate peak in TS(fq) df, // Set zero plane zo, // Set zero plane	• puteru(0.);	// Surface reflection coefficient	#include "nsc	pg10b.h"	Xia.
// set the initial frequency FLI_TYPE // set the delta frequency range. // Calculated resonance fq of spherical cloud // set the delta frequency theta, // Setatering angle a, // Setatering angle mu, // Surface reflection coefficient -i< mm < 0 phi, // Surface reflection coefficient -i< mm < 0 phi, // Surface reflection coefficient -i< mm < 0 phi, // Surface frezing angle ffills, // Minimum fq used to calculate peak in TS(fq) ffills, // Change in fq used to calculate peak in TS(fq) df, // Set zero plane zo, // Set zero plane	_puta(0.1 );	// sec inicial cloud regius			Ret,
// Galculated resonance fg of spharical cloud // set the delta frequency range. // Wesenmoer in host fluid == 2 Pi fg/Cl thata, // Settering angle a, // Gloud Radius mu, // Gloud Radius mu, // Surface reflection coefficient -i< mm < 0 phi, // Surface reflection coefficient -i< mm < 0 phi, // Surface reflection coefficient -i< mm < 0 phi, // Surface reflection coefficient -i< mm < 0 phi, // Surface reflection coefficient -i< mm < 0 phi, // Surface reflection coefficient -i< mm < 0 phi, // Surface reflection coefficient -i< mm < 0 phi, // Ghange in fg used to calculate peak in TS(fq) df, // Ghange in fg used to calculate peak in TS(fq) df, // Set zero plane zo, // Set zero plane	-putf(300);	// set the initial frequency	FLT_TYPE		Arb, Art,
// set the delta frequency free, // Calculated resonance of of sparital cloud theta, // Weenumber in host fluid == 2 Pi fqCl theta, // Sextering angle m, // Surface reflection coefficient -i< mu < 0 phi, // Surface Craing angle ffill, // Minimum fq used to calculate peak in TS(fq) ffill, // Minimum fq used to calculate peak in TS(fq) ffill, // Minimum fq used to calculate peak in TS(fq) ffill, // Ghange in fq used to calculate peak in TS(fq) ffill, // Set zero plane	xxx= putminmaxf(1fo(100	().); // Mex and Minimum frequency ran			Rei
theta, // Scattering angle  a, // Cloud Radius  an, // Surface reflection coefficient -ic mu < 0  phi, // Surface Grazing angle  ###, // Winiams fq used to calculate peak in TS(fq)  ###, // Maximus fq used to calculate peak in TS(fq)  ###, // Change in fq used to calculate peak in TS(fq)  ###, // Set zero plane  ###, // Set zero plane  ###, // Set zero calculate peak in TS(fq)  ####, // Set zero plane	-putdf(100.);	// set the delta frequency	ires,	// Calculated resonance Iq or spherical cloud // Wavenumber in host fluid == 2 Pi fq/Cl	ner, Res,
an, // Cloud Radius an, // Surface reflection coefficient -ic mu < 0 phi, // Surface reflection coefficient -ic mu < 0 phi, // Surface Greating angle fHIM, // Minimum fq used to calculate peak in TS(fq) fMAM, // Maximum fq used to calculate peak in TS(fq) df, // Change in fq used to calculate peak in TS(fq) zo, // Set zon plane			theta,	// Scattering angle .	Rre,
ma, // Surface reflection coefficient -1< mu < 0 phi, // Surface reflection coefficient -1< mu < 0 phi, // Surface Grazing angle ffHI, // Kiniama fq used to calculate peak in TS(fq) ffHI, // Maximum fq used to calculate peak in TS(fq) df, // Change in fq used to calculate peak in TS(fq) zo, // Set zon plane	. (0 8 0).		i	// Cloud Radius	Rab,
phi, // Surface Craing angle fMIN, // Minimum di used to calculate peak in TS(fq) fMAX, // Minimum fq used to calculate peak in TS(fq) df, // Change in fq used to calculate peak in TS(fq) df, // Change in fq used to calculate peak in TS(fq) cx, zzz,i.,0.376,Pl,0); Sec zero peak in TS(fq) cx, zzz,i.,0.376,Pl,0);	combrex (0.0°0);		· nu	// Surface reflection coefficient -1< mu < 0	Theta_st,
tHin, // Hinimum fq used to calculate peak in TS(fq)  fMX, // Haximum fq used to calculate peak in TS(fq)  df, // Change in fq used to calculate peak in TS(fq)  zo, // Set zero plane	- (a) - (b) - (b) - (c)		ph1,	// Surface Grazing angle	Theta_sa,
fMAX, // Maximum fq used to calculate peak in TS(fq)  df, // Change in fq used to calculate peak in TS(fq)  zo, // Set zero plane	-mt2(89.):		THIN,	// Minimum fq used to calculate peak in TS(fq)	Theta_sr,
dr, // Canage in queed to Calculate peak in 19(1q) ZO, // Set zero plane	-14./0.376;		fHAX,	// Maximum fq used to calculate peak in TS(fq)	Theta_rt,
NO. // Coet Zene Datable	-TSanderson(xxx,zzz,1.,0.3	376,PI,0);	ďť,	// Change in iq used to calculate peak in IS(iq)	IDeta_rl,
/ Dind speed in m/s			z0.	// Cot grad plans	Ineta_ro,

Theta_si,	Rsbm sqrt( (Ys-Yb)*(Ys-Yb) + (Ks-Xb)*(Xs-Xb));	-10.*log10(norm( exp(I* hzSinTbetart)/Rrt
kzSinThetarb,		+mup*exp(I*kzSinThetari)/Rri
kzSinThetert,	Theta_st= atan( (Ys-Yt)/ (Xs-Xt));	
kzSinTheteri,	Theta_sr= atan( (Ys-Yr)/ (Xs-Xr));	
SCOrt,	Theta_rt= atan( (Yr-Yt)/ (Xr-Xt));	
SCCri,	Theta_ri= atan( (Yr-Yi)/ (Xr-Xi));	
*dnm	Theta_rb= atan( (Yr-Yb)/ (Xr-Xb));	
BeamWidth,	Theta_sem atan( (Ys-Ya)/ (Xs-Xa));	
C_Beam,	Theta_sb= atan( (Ys-Yb)/ (Xs-Xb));	TSff= 20.*log10(SCCrt/2.*a);
BSF_arg.	Theta_si" atan( (Ys-Yi)/ (Xs-Xi));	
freq,		TS = TSff -DP_corr;
Съевш	kzSinThetarb=k*(Y1-Yt)*sin(Theta_rb);	
BSF,	kzSinThetart=ke(Yt-Yb)*sin(Theta_rt);	return (TS);
DP_corr,	hzSinThetari=k*(Yi-Yb)*sin(Theta_ri);	
TSff.		
TS,	SCCrt- SCC(ka,g,h,PI-Theta_rt,M);	
rsqmi, isqui, rmi, imi, irmi;	SCCri- SCC(ka,g,h,PI-Theta,ri,M);	
complex mi,sqmi;		***************************************
		extern "C"
mi=complex(-1.e0,0.);	mup = mu*SCCri*SCCri/SCCrt/SCCrt;	
rmi-real(m1);		woid SenecalSdepthlist(FLL_TYPE fq, FLL_TYPE cloudrad, FLL_TYPE sdepths, long
iml=imag(ml);	freq= kecl/2/Pl;	
irmi=imi/rmi;	BeamWidth=8.5*PI/180.;	int ind;
		FLT_TYPE w.k.g.z.dummy;
sqm1= sqrt(m1);	BeamWidth= PI/180.*(14.483 + -2.228*log10(freq));	
raqmi=real(sqm1);	<pre>C_Beam= sqrt((-log10(exp(1.))*BeamWidth*BeamWidth/4.)/log10(0.5));</pre>	complex h;
1sqm1=lmag(sqm1);	BSF_argw((Theta_st-Theta_sa)/C_Beam);	
		dummy* puta(cloudrad);
Ye= 60.96;	BSF= 20.*Logio(Rst/Rsa)	dummy= putf(fq); "
Yr=Ys+20.58000;	+10.*log10(*xp( BSF_arg*BSF_arg) );	w = 2*Plefq;
Yb= Ys+30.48000;		k = v/c1;
Yi- 2+Yb -Yt;		
Ya= Ye+26.52000;	#1fdef FARFIELD_APPROX	/* replace each depth with its TS e/
Xs= 0.;	// Fax-Field Approximations.	for (ind=0;ind <num;ind++)< td=""></num;ind++)<>
Xr= Xs+28.72000;		<b>~</b>
Xt=X1=Xb=Xa= Xs+37.01000;	Rrt - Rri - Rrb;	dummy putf(depths[ind]);
	kzSinThetart= 0.;	
Rat= sqrt( $(Ys-Yt)*(Ys-Yt) + (Xs-Xt)*(Xs-Xt));$	kzSinTheteri- kzSinThetarb;	
<pre>Rrb= sqrt( (Yr-Yb)*(Yr-Yb) + (Xr-Xb)*(Xr-Xb));</pre>		
<pre>Rrt= sqrt( (Yr-Yt)*(Yr-Yt) + (Xr-Xt)*(Xr-Xt));</pre>	#endif	z=put2(87.7824);
<pre>Rri= sqrt( (Yr-Y1)*(Yr-Y1) + (Xr-X1)*(Xr-X1));</pre>		-
Rar= sqrt( (Yr-Ya)*(Yr-Ys) + (Xr-Xs)*(Xr-Xs));	DP_corr = 20.*log10(1./Rra)	g = rhom()/rhol();
Rss= sqrt( (Ya-Ys)*(Ya-Ys) + (Xa-Xs)*(Xa-Xs));		h = cm(v)/c1;
Rra= sqrt( (Yr-Ya)*(Yr-Ya) + (Xr-Xa)*(Xr-Xa));		

dummy= SenecaTS(k,a,g,h,depths[ind],H);	PhaseSpeed3.1.2.cxx	•
depths[ind]=dummy;	CONTROL MAINTENANCE CONTROL	cl*CL;
		RECALC=1;
	***************************************	return(cl);
^	// PhaseSpeed (Phase SpeedCollective Oscillations Program) **	
Cash of short and short	***************************************	
ALGERTAN SEPTEM CARGO CA	"	
		•
-		
ביין וונך מח"פההרסעונדי"וונד סוניין יינד אדי	// volume fraction. Future versions of this code will be modified to	
	// include many bubble radii and perhaps continuous distributions	
( ) Commute Intermity Reflection Coefficient for rough mirface	// of bubbles.	
// heard on Pierson-Moskowitz Wave Height Spectrum and the	"	
/ Kitchoff Browning and Con	Based	
// Attendit Approximation.	// liquids:Comparison between theory and experiments.	•
FLT TYPE	"	extern "G"
		FLT_TYPE putBetaList(FLT_TYPE betalist[], long num)
- Protection	"	-
6008;		
•	Winclude "nscsp3.1.h"	INT_TYPE
//Mean-Square Wave Height from Pierson-Wuskowitz Spectrum	#include "PhaseSpeedGlob.h"	index; // Number of bubble radii in distribution
	A STREET OF THE STREET	
bagrd = ALPHA*U*U*U*U*(4.*BETA*G*G);	#10clude decalo.n.	
	Winclude <atdlib.b></atdlib.b>	beta = betallet[0];
ans = - exp( - 2. *hsqrd*kz*kz ) ; //2 for amplitude refl	#include <math.b></math.b>	
//4 for intensi		Beta_Amp=0.;
		NUMDIST - num/5;
return aus:	#define SIGMA ((1.E-5)/0.01*(75.68-0.138*TMP-3.56*-6*TMP**TMP*4.74-7*IMP*TMP))	INP*IMP))
	#define Garma 1.4	for (index=0;index < NUMDIST;++index)
	#define EPS 1.E-6	-
		numbub[index] = betalist[5eindex+1];
		bubred[index] = betalist[5=index+2];
Sunday ALPHA		bubradad[index] = betalist[5*index+3];
	extern "C"	bubradmin[index] = betalist[5eindex+4];
S S S S S S S S S S S S S S S S S S S	char* INFO_PhaseSpeed()	bubradmax[index] = betalist[5eindex+5];
	#	
	TOTAL TRANSPORTED TIPE CART TOO PY Department of DINAPI	
extern "C"		
FLI_TYPE get_mu_approx( FLI_TYPE U, FLI_TYPE kz )		RECALC-1;
FIL TYPE ans;		return beta;
ans-mu_approx(U,kz);	***************************************	
return and		
	extern "C"	
	FLT_TYPE putc1(FLT_TYPE CL)	

```
#define KAPPA KO*TK/TO*sqrt(TK/TO)*(TO+TA/ *xp(TB/TO))/(TK+TA/*xp(TB/TK))
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             //Kappa == Thermal Conductivity see Pierce, pg 513, eq 10-1.16b
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             complex PHI(FLI_TYPE v,FLI_TYPE r,FLI_TYPE Po)
- 3.56e-6*T2 + 4.7e-7*T2*tmp));
                                                                                                                                                                                                                              ans*(dyne/CM*(75.68 - 0.138*tmp
                                                                                               // Surface tension of bubble
                               FLI_TYPE Signa (FLI_TYPE tmp)
                                                                                                                               FLI_TYPE dyne, CM, T2, ans;
                                                                                                                                                              dyne = (1.E-5);
CM = .01; // centimeter
                                                                                                                                                                                                                                                                                                                                              #define KO 2.64529e-2
                                                                                                                                                                                                                                                                                                                                                              #define TO 273.15
#define TK (TMP+TO)
                                                                                                                                                                                                                                                                                                                                                                                              #define TA 245.4
#define TB 27.6
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sqrt_1_chi,
                                                                                                                                                                                                 T2-tmp*tmp;
                                                                                                                                                                                                                                                                                 return ans;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Dtherm;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               I_chi;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               ans,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               complex
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               cht,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                Po = 1.e5 + rhol() * 9.81 * zz;
                                                                                                                                               extern "C"
FLI_TYPE putZ(FLI_TYPE ZZ)
                                                                                                                                                                                                                                                                                                                                                              extern "C"
FLI_TYPE putf(FLI_TYPE F)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               return( rhol0 );
                                                                                                                                                                                                                                                                                                                                                                                                                                             //K = 2*PI*f/cl;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             FLT_TYPE rhol()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               RECALC-1;
                                                                                                                                                                                                                                                                                return(zz);
                                                                                                                                                                                                                                                                                                                                                                                                                                                               return(f);
                                                                                return(Po);
                                                                                                                                                                                                                                              RECALC=1;
                                                RECALC=1;
                                                                                                                                                                                                                :ZZ=ZZ
                               Po=PO;
                                                                                                                                                                                                                                                                                                                                                                                                                               f=F:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              return( rhog0*Po/(1.E5) *273.12/(273.12+TMP));
                                                                                                                                                                                                                                                                                                                                                                                                                               //
                                                                                                                                                                                                 return( (1.-beta)erhol() +betaerhog());
}
                                                                                                                                                                                                                                              extern "C"
FLI_TYPE putrhog(FLI_TYPE RHGG)
               FLT_TYPE putrhol(FLT_TYPE REGL)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                FLI_TYPE putPo(FLI_TYPE PO)
                                                                                                                                                                                                                                                                                                                                                                 return(rhogo);
                                                                                                                return(rhol0);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              FLI_TYPE rhom()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                FLT_TYPE rhog()
                                                                                                                                                                                                                                                                                                             rhogo-RHOG;
                                                               rhol0-RHOL;
                                                                                                                                                                                                                                                                                                                                 RECALC-1;
                                                                                RECALC-1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 exterm "C"
 extern "C"
```

```
complex CompSimpsonInt( complex (*func)(FLI_TYPE), FLI_TYPE a, FLI_TYPE b)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sum=func(a); // special case for delta fn.
                                                                                                                                                                                                                                                                                  #define REAL_F(x) ( real( (*func)(x) ) ) #define IMAG_F(x) ( imag( (*func)(x) ) )
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 for (1=0;1<MAXITTER; ++1)
{</pre>
                                                                                                                                                                                                                                                                                                                                                                                                                                          complex sum;
FLI_TYPE realint,imagint;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            sum += (func(x));
x += dx;
sum += func(x);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       FLT_TYPE resum, insum;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           dx = (b-a)/HAXIIIER;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  INT_TYPE 1, MAXITTER;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        resum= real(sum);
imsum= imag(sum);
                    x += dx;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              FLT_TYPE x,dx;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             MAXITTER-256;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    sum ** dx;
                                                                                    * dx;
                                                                                                                                                     return sum;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        if (dx==0)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  10 - una
                                           ::
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             FLI_TYPE Simpsonint(FLI_TYPE (*func)(FLI_TYPE), FLI_TYPE a, FLI_TYPE b)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   ensequt( pb/rhol()/r/r*( real(PHI(w.r.Po))- 2*SIGMA/r/pb));
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             +imag(PHI(u,r,Po))*po(r,Po)/(2*r*r*rhol()*u)));
                                                                                                                                                                                                                                                                                      ans=( r*w*w/(2*cl) + (2*MU/(r*r*rhol())
                                                                                                                                                                                                                                                                                                                                              PLI_TYPE wo(FLI_TYPE r,FLI_TYPE w)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          for (1=0;i<MAXITTER; ++i)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              INT_TYPE 1, MAXITTER;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       dx = (b-a)/MAXITIER;
                                                                                      HU = 1.02* (1.E-3);
                                                                                                                                                                                                                                                                                                                                                                                               FLT_TYPE ans, pb;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        sum-func(a);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        FLT_TYPE x,dx;
                                                                                                                                                                                                                                                                                                                                                                                                                                          pb-po(r,Po);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     FLT_TYPE sum;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       MAXITTER-256;
                                                                                                                                                                                                  return ans;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  return ans;
                    { FLT_TYPE NU;
                                           complex ans;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  11 (dr==0)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              :0 = wns
                                                                                                                                                                                                                                                                                                                                 extern "C"
                                                                                                      //Dtherm-Kappa/rhogO/Op == See Prosperetti-Thermal Behavior of oscillating
// gas bubbles. JFN '91(222)
// Same ss (1-Gamma)/Gamma-TK/Po +
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              ans=( Po + 2*SIGMA/x); // pressure inside the bubble
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              complex b(FLT_TYPE w,FLT_TYPE r,FLT_TYPE Po,FLT_TYPE cl)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 Dtherm=((-1. + Gamma)*(TK)*KAPPA/(Gamma*Po));
                                                                                                                                                                                                                                                                                                                                                      (-1. + sqrt_I_chi*coth(sqrt_I_chi) )) );
                                                                                                                                                                                                                                                                                                                               answ(3.*Gamma/(1. - 3.*I*(-1. + Gamma)*chi*
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   FLI_IYPE po(FLI_IYPE r, FLI_IYPE Po)
                                                                                                                                                                                                                                                                                      sqrt_I_chi = sqrt(I/chi);
                                                                                                                                                                                                                                         chi=( Dtherm/w/r/r);
                      1f (Garma ==1.0)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     FLT_TYPE ans;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                return (ans);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             return ans;
                                           ans = 3;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       Fundef KAPPA
                                                                                      *18* {
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           Sundef TO
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               Nundef TA
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    #undef TB
                                                                                                                                                                                                                                                                                                                                                                                                                     ;;
```

```
tempdisp += 4*PI*cl*cl*numbub[i]* Beta_Amp *
CompSimp*onInt(Dispintagrand,a,b);
                                                                                                                                                                                                                                                                                                                                                                                                                                      if (Beta_Amp==0)
Beta_Amp = beta/ check_Beta();
                                                                                                                                                                                                                                                                                                                   complex denom, tempdisp, WO; FLI_TYPE a, b, redisp, imdisp;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     tempdisp = complex(0,,0.);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            for (1=0; 1<NUMDIST; ++1)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             indispeimag(tempdisp);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         redispersal(tempdisp);
imans - imag(ans);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                if (a < 0.) a=EPS;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    a = bubradmin[1];
b = bubradmax[1];
                                                                             reans = real(ans);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    return(tempdisp);
                                                                                                 imans= imag(ans);
                                                                                                                                                                                                                                                            complex dispersion()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  { BubIndex=1;
                                                                                                                                                          return ans;
                  <u>:</u>
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    Ξ
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    ans = x *exp( - arg*arg)/DispDenom(x);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 ans = bubrad[1]/DispDenom(bubrad[1]);
                                                                                                                                                                                                                                                                                                                                                                                                                                        betavalue += SimpsonInt(BETA,a,b);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                arg * (x-bubrad[1])/bubradsd[1];
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   complex Dispintegrand(FLT_TYPE x)
                                      return 4*PI/3*numbub[i]*ans;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            FLI_TYPE arg, reans, inens;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           reans= real(ans);
                                                                                                                                                                               int i;
FLI_TYPE betavalue,a,b;
                                                                                                                                                                                                                                                                             for(1=0; 1<NUMDIST;++1)
                                                                                                                                                                                                                                                                                                                  BubIndex = 1;
a=bubradmin[i];
b=bubradmax[i];
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              1f (bubraded[1]==0)
                                                                                                                  FLT_TYPE check_Beta()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       return betawalue;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         i - Subinder;
                                                                                                                                                                                                                                          betavalue=0;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            complex sus;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  INT_TYPE 1;
                                                                                                                                                                                                                                      arg = (x-bubrad[i])/bubraded[i];
ans = $ *x*x * exp( - arg * arg);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                              ans = (-w+w +2*I+w*b(w,x,Po,c1) + W0*W0);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           ans = bubrad[i]*bubrad[i]*bubrad[i];
                                                                                                                                                                                                                                                                             complex DispDenom(FLT_TYPE x)
                                                                                                                                                                                                                                                                                                                                                            FLT_TYPE v,reans,imans, WO;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                FLI_IYPE BETA(FLI_IYPE x )
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         if (bubraded[1]==0)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   reans-real(ans);
imens-imag(ans);
                                                                                                                                                                                                                                                                                                                                                                                                                                        Wo = wo(x,w);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     FLT_TYPE ans;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         FLI_TYPE arg;
                                                                                                                                                                                                                                                                                                                                         complex sns;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                return ans;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             INT_TYPE 1;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      1-SubIndex;
                                                                                                                                                                                                                                                                                                                                                                                                   w = 2ePIef;
                                                                                                                  #undef REAL_F
                                        return sum;
```

```
wold CaFQlist (FLT_TYPE *freqs; FLT_TYPE *calistt, long num)
                                                                                                                                                                                                                                                                                                                                                                   // returns complex resonance fq of bubble
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               FLT_IYPE GETWo(FLT_IYPE rad, FLT_IYPE fq)
                                             complex sqrtlplusdispersion;
FLT_TYPE getcm(FLT_TYPE frq)
                                                                                                                                                                      tempon=cl/real(clom(w));
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         FLI_TYPE dummyfq,W0;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      WD = wo(rad,2*PI*f);
                                                                      FLT_TYPE tempom, w;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         dummyfq-putf(fq);
                                                                                                                                                                                                                      return(tempon);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   dummy, relem;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    return(WO);
                                                                                                                      w-2.PIefrq;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        extern "C"
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          ipq;
                                                                                                                                                                                                                                                                                                                                                                                                                   extern "C"
                                                                                                                                                                                                                                                                                                                                                                                                                     // tempcmesqrt( Posgarmma/cl/cl+beta*(1.-bsta) *rhol());
// (Po *garmma/cl/cl+beta*(1.-bsta) *rhol()));
                                                                                                                                                                                                                                            rclcmlist=real(((complex * ) clcmlist)[0]);
rclcmlist=real(((complex * ) clcmlist)[3]);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  // garmme=1./3. * real(PHI(f,bubrad[0],Po));
                                               rclcmlist=real( clcmlist[jj]); iclcmlist=imag( clcmlist[jj]);
                           clemlist[jj]=clem(w);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      complex cm(FLI_TYPE w)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      tempcurcl/clcm(w);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            return( tempon);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             complex tempon;
                                                                                                                                                                                                                                                                                                                     RECALC=0;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                extern "C"
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        clemist = (complex *) malloc( (size_t) NUMFQS * sizeof(complex));
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (Po *garmma/cl/cl+beta*(1.-beta) *rhol());
  complex tempclcm, sqrtiplusdispersion, garmus;
                                                                                                                           complex tempolom, sqrt1plusdispersion, garmma;
                                                                                                                                                                                                 sqrtiplusdispersion*sqrt(1* dispersion());
                                                                                                                                                                                                                                                                                                                                                                                                                   // garmma=1./3. * real(PHI(f,bubrad[0],Po));
                                                                                                                                                                                                                                            if(aqrtiplusdispersion!"complex(0.,0.))
tempclcm=conj(sqrtiplusdispersion);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 FLI_TYPE w,relemist,iclemist;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     w-2*PI*putf(fqlist[jj]);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   for (jj=0;jj<NUMFQS;jj++) {
                                                                                                                                                                                                                                                                                                                                                                                                                                                                     // tempomesqrt( Poegarmas/
                                                  complex clcm(FLT_TYPE w)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               1f (RECALC=-1) {
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               free(clcmlist);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     return( tempclcm);
                                                                                                                                                                                                                                                                                                                                              tempolom-CIMF;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           word CalcClCm()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               int jj;
```

```
// Sound speed in surrounding medium // density of surrounding medium // density of gas medium
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    // set the initial frequency
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                // Ambient Surrounding pressure
                                                                                                                                                                                           // dummy variables
                     complex jsqrt(const complex& z) ;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  putBetalist(BETALIST,6);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         xxx= putcl(1500.);
xxx= putrhol(1000.);
xxx= putrhog(1.);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           BETALIST[2]=.0005;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             BETALIST[3]=0.;
BETALIST[4]=0.;
BETALIST[5]=0.;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  BETALIST[0] -. 001;
                                                                                extern "C"
void initPhaseSpeed()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       BETALIST[1]-1.;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              xxx= putZ(0.);
                                                                                                                                                                     BETALIST[16],
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  xxx=putf(0.1);
                                                                                                                                                                                                                                                                                                                                                                                                                                                 NUMBER - 0;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                     RECALC - 0;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            NUMPOS-0;
                                                                                                                                               FLT_TYPE
                                                                                                                                                                                                                                                                                                 INT_TYPE
                                                                                                                                                                                                                                                            :000
                                                                                                                                                                                                                                                                                                                   dumny= 8.68569= w* imag( clemiist[ind])/cl/100; // Attenuation(freqs[ind])ii; Attenlist[ind]=dumny;
                                                                                                                                                                                                                                      char.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    /* replace each frequency with its Polytropic Index */
                                                                                                                              /* replace each frequency with its Attenuation */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       word PolytropicIndex( FLT_TYPE *freqs, long num)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     w=putf(freqs[lad]);
dummy= 1./3. * real(PHI(w,bubrad[0],Po]);
freqs[lad]=dummy;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         for (ind*0;ind<NUMFQS;ind++)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                for (ind=0;ind<num;ind++)
                                                                                                                                                                                                                                                                           w=2.PI.fqlist[ind];
                                                                                                                                                                                           Calcilon();
                                                                 dummy, w;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            FLT_TYPE
                                           FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              exterm "C"
                   ipu;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            ipq;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   wold AttenFQ11st( FLI_TYPE *freqs, FLI_TYPE *Attenlist, long num)
                                                                                                                                                                                                                                                                                                                                                                                                                                                 /* replace each frequency with its Cm */
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   ans = 8.68589* u* imag(1./cm(u))/100;
                                                                                                                                                                                                                                    dummy= cl/real(clcmlist[ind]);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                extern "C"
FLI_TYPE Attenuation(FLI_TYPE freq)
                                                                                                                                                                                                                                                                                                                     dummy= cl/real( clcmlist[3]);
dummy= cl/real(clcmlist[0]);
                                                                                                                                                                     for (ind=0;ind<FUNFQS;ind++)
                                                                                                    rclcm=cl/real(clcmlist[0]);
rclcm=cl/real(clcmlist[1]);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   w* 2*PI*putf(freq); .
                                                                                                                                                                                                                                                          cmlistt[ind] -dummy;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              return ans;
                                                           CalcClCm();
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 FLT_TYPE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          ans.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            extern "C"
```

beta, numbub [MARBUBDIST], bubrada [MARBUBDIST], bubrada [MARBUBDIST], bubrada [MARBUBDIST], bubrada [MARBUBDIST], strang, //\*cmlist-WILL, //\*ttenlist-WILL, \*qlist-WILL,

PhaseSpeedGlob.cxx

#include "nscsp3.1.h" #include "PhaseSpeedGlob.h"

FLT\_TYPE

BubIndex, NUMDIST,

cl. Po. zz. rholO, f.

RECALC, NUMFQS;

complex

\*clcmlist=WULL;

PhaseSpeedMain3.1.c

complex.h

## **Bibliography**

Abramowitz, M., & Stegun, I.A. 1972. Handbook of mathematical functions. New York: Dover.

Adair, R., Huster, M., & Miklovic, D. 1992. Backscattering of underwater noise by bubble clouds. J. Acoust. Soc. of Amer., 92, 2479.

Adair, R.G., & Huster, M.E. 1993. High resolution surface reverberation imaging during CST7 Phase 2. Tech. rept. ARS-239-001-TR. Areté Associates, La Jolla, CA.

Allen, W.T.R. 1983. Wind and Sea. Canadian Government Publishing Center.

Anderson, Victor C. 1950. Sound Scattering from a fluid sphere. *J. Acoust. Soc. of Amer.*, **22**(4), 426–431.

Bannister, R., Denham, R., Guthrie, K., & Browning, D. 1981. Measurements of the low-frequency wind generated ambient noise in the deep ocean. *J. Acoust. Soc. of Amer.*, **69**, 595.

Bowman, J. J., Senior, T. B. A., & Uslenghi, P. L. E. (eds). 1987. *Electromagnetic and Acoustic Scattering by Simple Shapes*. Revised edn. New York: Hemishphere Publishing.

Brekhovskikh, L. M., & Lysanov, Y. P. 1991. Fundamentals of Ocean Acoustics. second edn. Wave Phenomenon. New York: Springer-Verlag.

Brennen, C.E. 1995. Cavitation and bubble dynamics. New York: Oxford University Press.

Carey, W. M., & Browning, D. 1988. Low-frequency ocean ambient noise: Measurement and theory. Pages 361-76 of: Kerman, B. R. (ed), Sea Surface Sound: Natural mechanisms of surface generated noise in the ocean. Boston: Kluwer.

Carey, W. M., & Fitzgerald, J. W. 1993. Low frequency noise from breaking waves. *Pages 277–304 of:* Kerman, B. R. (ed), *Natural Physical Sources of Underwater Sound: Sea Surface Sound*, vol. 2. Boston: Kluwer.

Carey, W. M., & Monahan, E. C. 1990. Guest Editorial. *IEEE J. Ocean Eng.*, **15**(4), 265–267.

Carey, W. M., & Roy, R. A. 1993. Sound scattering from microbubble distributions near the sea surface. *Pages 25–43 of:* Ellis, D., Preston, J. R., & Urban, H. G. (eds), *Ocean Reverberation*. Boston: Kluwer.

Carey, W. M., & Wagstaff, R. A. 1986. Low frequency noise fields. J. Acoust. Soc. of Amer., 80(5), 1523–1526.

Carey, W. M., Fitzgerald, James W., Monahan, E. C., & Wang, Q. 1993. Measurement of the sound produced by a tipping trough with fresh and salt water. J. Acoust. Soc. of Amer., 93(6), 3178–3192.

Carey, W.M., & Bradley, M.P. 1985. Low-frequency ocean surface noise sources. J. Acoust. Soc. of Amer., 78(S1–S2).

Chapman, R. P., & Harris, J. 1962. Surface backscattering strength measured with explosive sound sources. *J. Acoust. Soc. of Amer.*, **34**, 1592–97.

Chapman, R.P., & Scott, H.D. 1964. Surface backscattering strengths measured over an extended range of frequencies and grazing angles. *J. Acoust. Soc. of Amer.*, **36**, 1735–1737.

Cheyne, S.A., Stebbings, C.T., & Roy, R.A. 1995. Phase velocity measurements in bubbly liquids using a fiber optic laser interferometer. *J. Acoust. Soc. of Amer.*, **97**(3), 1621–4.

Clay, C.S., & Medwin, H. 1977. Acoustical Oceanography: prinicples and applications. New York: John Wiley & Sons.

Commander, K. W., & Prosperetti, A. 1989. Linear pressure waves in bubbly liquids: Comparison between theory and experiments. *J. Acoust. Soc. of Amer.*, 85(2), 732–746.

Crighton, D. G., Dowling, A. P., Ffowcs Williams, J. E., Heckl, M., & Leppington, F. G. 1992. *Modern Methods in Analytical Acoustics: Lecture Notes*. New York: Springer-Verlag.

d'Agostino, L., & Brennen, C. E. 1983. On the acoustical dynamics of bubble clouds. ASME Cavitation and Multiphase Flow Forum, 72–75.

Donelan, M. A., Hamilton, J., & Hui, W. H. 1985. Directional spectra of wind generated waves. *Philos. Trans. R. Soc. London Ser. A*, **315**, 509–562.

Eckart, Carl. 1953. Sea surface scattering. J. Acoust. Soc. of Amer., 25, 566–570.

Farmer, D., & Vagle, S. 1989. Waveguide propagation of ambient sound in the ocean–surface bubble layer. J. Acoust. Soc. of Amer., 86, 1897–1908.

Farmer, D. M. 1992 (March). *FLEX Instrument Survey*. Tech. rept. University of Brittish Columbia.

Flax, L., Gaunaurd, G.C., & Überall, H. 1981. Physical Acoustics: principles and methods. Vol. XV. New York: Academic Press. Chap. 3.

Fox, F.E., et al. 1955. Phase velocity and absorption measurements in water containing air bubbles. J. Acoust. Soc. of Amer., 27, 534–539.

Furduev, A.V. 1966. Undersurface cavitation as a source of noise in the ocean. *Atmospheric and Oceanic Physics*, **2**, 523.

Gauss, R.C., Fialkiowski, J.M., & Soukup, R. J. 1993. Low–frequency direct—path surface and volume scattering measured using narrowband and broadband pulses. *Page 235 of:* Ellis, D., *et al.* (eds), *Ocean Reverberation*. Boston: Kluwer.

Gilbert, K. E. 1993. A stochastic model for scattering from the near-surface oceanic bubble layer. J. Acoust. Soc. of Amer., 94(6), 3325-34.

Gouse, S.W., & Brown, G.A. 1964. A survey of the velocity of sound in two-phase mixtures. Tech. rept. Paper 64-WA/FE-35. ASME.

Gragg, R. F., & Wurmser, D. 1993. Low-frequency scattering from intermediate bubble plumes: theory and computational parameter study. *J. Acoust. Soc. of Amer.*, **94**(1), 319–29.

Hasselman, K., Ross, d. B., Müller, P., & Sell, W. 1976. A parametric wave prediction model. *J. Phys. Oceanogr.*, **6**, 200–228.

Henyey, F. S. 1991. Acoustic scattering from ocean microbubble plumes in the 100 Hz to 2 kHz region. J. Acoust. Soc. of Amer., 90(1), 399-405.

Hollet, R. D. 1994. Observations of underwater sound at frequencies below 1500 Hz from breaking waves at sea. J. Acoust. Soc. of Amer., 95(6), 165–70.

Hollet, R.D., & Heitmeyer, R.M. 1988. Noise generation by bubbles formed in breaking waves. *Pages 449–461 of:* Kerman, B. R. (ed), *Sea Surface Sound:* Natural mechanisms of surface generated noise in the ocean. Boston: Kluwer.

Karplus, H.B. 1958. The velocity of sound in a liquid containing gas bubbles. Tech. rept. COO-248. Illinois Inst. Tech.

Keller, J. B., & Kolodner, I. I. 1956. Damping of underwater explosion bubble oscillations. *J. Appl. Phys.*, 27, 1152–1161.

Kennedy, R.M., & Goodnow, T.V. 1990. Measuring the vertical directional spectra caused by sea surface sound. *IEEE J. Ocean Eng.*, **15**(4), 299–310.

Kerman, B.R. 1984. Underwater sound generation by breaking waves. J. Acoust. Soc. of Amer., 75, 148.

Kewley, D. J., Browning, D. G., & Carey, W. M. 1990. Low-frequency wind-generated ambient noise source levels. *J. Acoust. Soc. of Amer.*, 88, 1894–1902.

Kibblewhite, A.C., Shooter, J.A., & Watkins, S.L. 1976. Examination of attenuation at very low frequencies using the deep water ambient noise field. *J. Acoust. Soc. of Amer.*, **60**, 1040.

Kinsler, L. E., Frey, A., Coppens, A. B., & Sanders, J. V. 1982. Fundamentals of Acoustics. New York: John Wiley & Sons.

Knudsen, V.O., Alford, R.S., & Emling, J.W. 1948. Underwater Ambient Noise. J. Mar. Res., 7, 410.

Kolaini, A. R., Roy, R. A., Crum, L. A., & Mao, Yi. 1993. Low-frequency underwater sound generation by impacting transient cylindrical water jets. *J. Acoust. Soc. of Amer.*, **1994**(5), 2809–20.

Kolaini, A.R., Roy, R.A., & Gardner, D.L. 1994. Low-frequency acoustic emissions in fresh and salt water. *J. Acoust. Soc. of Amer.*, **96**(Sep), 1766–72.

Koller, D. P., & Shankar, P. M. 1994. Acoustical emissions from bubble clouds. *Ultrasonics*, **32**(3), 229–33.

Koller, D.P., & Shankar, P.M. 1993. Low-frequency oscillations of bubble plumes. J. Acoust. Soc. of Amer., 93(3), 1362–1364.

Konrad, W. L., & Geary, J. III. 1991. Potential application fo a parametriec source for target strength measurements. Tech. rept. TM 811067. Naval Underwater Systems Center.

Lamarre, E. 1993. An experimental study of air entrainment by breaking waves. Ph.D. thesis, Massachusetts institute of Technology & Woods Hole Oceanographic Institution, MA.

Lamarre, E., & Melville, W. K. 1992. Instrumentation for the measurement of void-fraction in breaking waves: laboratory and field results. *IEEE J. Ocean Eng.*, 17(2), 204–15.

Lamarre, E., & Melville, W. K. 1994. Sound–speed measurements near the ocean surface. J. Acoust. Soc. of Amer., 96(6), 3605–16.

Lighthill, James (ed). 1975. Waves in Fluids. Cambridge University Press.

Longuet-Higgens, M. 1993. Bubble noise mechanisms—a review. *Pages 419–452 of:* Kerman, B. R. (ed), *Natural Physical Sources of Underwater Sound:* Sea Surface Sound, vol. 2. Boston: Kluwer.

Lu, N. Q., Prosperetti, A., & Yoon, S. W. 1990. Underwater noise emissions from bubble clouds. *IEEE J. Ocean Eng.*, **15**, 275–281.

McDaniel, S. T. 1988. High-frequency sea surface scattering: Recent progress. J. Acoust. Soc. of Amer., 84.

McDaniel, S. T. 1993. Sea surface reverberation: a review. J. Acoust. Soc. of Amer., 94(4), 1905–22.

McDonald, B. E. 1991. Echoes from vertically striated subresonant bubble clouds: A model for ocean surface reverberation. *J. Acoust. Soc. of Amer.*, **89**, 617–22.

Miklovic, D. W., Huster, M. E., & Moser-Breaker, Michelle. 1995. The spatial variability of surface reverberation under high and low wind speeds. *IEEE J. Ocean Eng.*, October, Accepted for publication. Accepted for publication.

Minneart, M. 1933. On musical air–bubbles and the sounds of running water. *Phil. Mag.*, **16**, 235–248.

Monahan, E. C. 1988. Whitecap coverage as a remotely monitorable indication of the rate of bubble injection into the oceanic mixed layer. *Pages 85–96 of:* Kerman, B. R. (ed), *Sea Surface Sound*. Boston: Kluwer.

Monahan, E. C., & Lu, M. 1990. Acoustically relevant bubble assemblages and their dependence on meteorological parameters. *IEEE J. Ocean Eng.*, **15**(4), 340–9.

Monahan, Edward C., & Niocaill, Gearoid Mac (eds). 1986. Oceanic whitecaps and their role in air-sea exchange processes. Kluwer.

Morris, G.B. 1978. Depth dependence of ambient noise in the northeastern Pacific Ocean. J. Acoust. Soc. of Amer., 64, 581–590.

Morse, P., & Ingard, U. K. 1968. *Theoretical Acoustics*. New York: McGraw-Hill.

Morse, P.M., & Feshbach, H. 1953. *Methods of theoretical physics*. New York: McGraw-Hill.

NDRC. 1946. The physics of sound in the sea. Tech. rept. National Defense Research Committee., Washington, DC. United States. Office of Scientific Research and Development.

Nicholas, M., Roy, R. A., Crum, L. A., Oğuz, H., & Prosperetti, A. 1994. Sound emissions by a laboratory bubble cloud. *J. Acoust. Soc. of Amer.*, **95**(6), 3171–82.

Nicholas, M., Roy, R. A., Crum, L. A., Carey, William M., & Schindall, J. to be submitted. Passive Underwater Acoustic Emissions from Artificial Bubble Clouds. *J. Acoust. Soc. of Amer.* 

Nützel. 1993. Measurements of acoustic backscattering of the near-surface layer. *In*: Ellis, D., Preston, J. R., & Urban, H. G. (eds), *Ocean Reverberation*. Boston: Kluwer.

Nützel, B., Herwig, H., Koenigs, P. D., & Monti, J. M. 1994. Acoustic backscatter measurements in the North Sea: 3–18 kHz. *J. Acoust. Soc. of Amer.*, **95**(5), 2488–94.

Oğuz, H. N. 1994. A theoretical study of low-frequency oceanic ambient noise. J. Acoust. Soc. of Amer., 95(4), 1895–1912.

Oğuz, H.N., Prosperetti, A., & Kolaini, A.R. 1995. Air entrapment by a falling water mass. *Journal of Fluid Mechanics.*, **294.** pp. **181-207. 10 July 1995.** boundary-integral-equations. bubbles. flow-simulation.

Ogden, P. M., & Erskine, F. T. 1994. Surface scattering measurements using broadband explosive charges in the Critical Sea Test experiments. *J. Acoust. Soc. of Amer.*, **95**(2), 746–61.

Pierce, Allan D. 1989. Acoustics: An introduction to its physical principles and applications. Woodbury, NY: Acoustical Society of America.

Pierson, W. J., & Moskowitz, L. 1964. A proposed spectral form of fully developed wind seas based on the similarity theory of S. A. Kitaigorodskii. *J. Geophys. Res.*, **18**, 5181–5190.

Prosperetti, A., Crum, L.A., & Commander, K.W. 1988. Nonlinear bubble dynamics. J. Acoust. Soc. of Amer., 83, 502–513.

Prosperetti, A., Lu, N. Q., & Kim, H. S. 1993. Active and passive acoustic behavior of bubble clouds at the oceans surface. *J. Acoust. Soc. of Amer.*, 93(6), 3117–3127.

Prosperetti, A. P. 1988a. Bubble Dynamics in ocean ambient noise. *Pages* 151–71 of: Kerman, B. R. (ed), *Sea Surface Sound*. Boston: Kluwer.

Prosperetti, Andrea. 1988b. Bubble-related ambient noise in the ocean. J. Acoust. Soc. of Amer., 84(3), 1042–54.

Prosperetti, Andrea. 1991. The thermal behavior of oscillating gas bubbles. J. Fluid Mech., 222, 587–616.

Prosperetti, A.P. 1985. Bubble related ambient noise in the ocean. J. Acoust. Soc. of Amer., 78(S1).

Prosperetti, A.P., & Lu, N.Q. 1988. Cavitation and bubble bursting as sources of oceanic ambient noise. *J. Acoust. Soc. of Amer.*, **84**, 1037–1041.

Pumphrey, H.C., & Ffowcs-Williams, J.E. 1990. Bubbles as sources of ambient noise. *IEEE J. Ocean Eng.*, **15**, 268–74.

Rayleigh, Lord (Strutt, J.W.). 1877. The theory of sound. London: MacMillan and Co.

Roy, R. A., Carey, W. M., Nicholas, M., Schindall, J., & Crum, L. A. 1992. Low-frequency scattering from submerged bubble clouds. *J. Acoust. Soc. of Amer.*, **92**(5), 2993–6.

Roy, R. A., Schindall, J., Carey, W. M., & Crum, L. A. in press. Can near–surface bubble clouds and plumes lead to anomalous perturbations in low–frequency sea–surface scattering?

Roy, R.A., Nicholas, M., Markiewicz, K., & Crum, L.A. 1991. Collective oscillations in fresh and salt water bubble clouds. *J. Acoust. Soc. of Amer.*, 2318.

Rschevkin, S. N. 1963. The Theory of Sound. New York: Pergamon Press.

Ruggles, A.E. 1987. The propagation of pressure perturbations in bubbly air/water flows. Ph.D. thesis, Rensselaer Polytechnic Institute, NY.

Sarkar, K., & Prosperetti, A. 1993. Backscattering of underwater noise by bubble clouds. J. Acoust. Soc. of Amer., 93(6), 3128–38.

Silberman. 1957. Sound velocity and attenuation in bubbly mixtures measured in standig wave tubes. *J. Acoust. Soc. of Amer.*, **29**, 925–933.

Spitzer, L. Jr. 1943. *NDRC Report No. 6*. Tech. rept. 1–sr20–918. National Defense Research Committee., Washington, DC. United States. Office of Scientific Research and Development.

Thorpe, S. A. 1982. On the clouds of bubbles formed by breaking wind—waves in deep water, and their role in air—sea gas transfer. *Phil. Trans. Roy. Soc.*, **304**, 155–185.

Thorpe, S.A. 1984. The effect of Langmuir circulation on the distribution of submerged bubbles caused by breaking waves. *J. Fluid Mech.*, **142**, 151–70.

Thorsos, Eric I. 1990. Acoustic scattering from a "Pierson-Moskowitz" sea surface. J. Acoust. Soc. of Amer., 88(1), 335–349.

Urick, R. J. 1967. Principles of Underwater Sound for Engineers. New York: McGraw-Hill.

Urick, R. J. 1984. Ambient noise in the sea. Washington, D.C.: Department of the Navy.

Urick, R.J., & Hoover, R.M. 1956. Backscattering of sound from the sea surface: Its measurement, causes, and application to the prediction of reverberation levels. *J. Acoust. Soc. of Amer.*, **28**, 1038–1042.

Vagle, S., & Farmer, D. M. 1992. The measurement of bubble-size distributions by acoustical backscatter. *Journal of Atmospheric and Oceanic Technology*, **9**(5), 630–44.

Wenz, G.M. 1962. Acoustic Ambient Noise in the Ocean: Spectra and Sources. J. Acoust. Soc. of Amer., 34, 1936.

Whittenborn, A. 1976. Ambient Noise and Associated propagations factors as a function of depth and wind speed in the deep ocean. Tech. rept. ADC006902. DTIC, Cameron Station.

Wilson, J.H. 1980. Low-frequency wind-generated noise produced by the impact of spray with the ocean surface. J. Acoust. Soc. of Amer., 68, 952.

Wood, A. B. 1941. A Text Book of Sound. New York, Macmillan.

Wu, J. 1981. Bubble populations and spectra in near–surface ocean: summary and review of field measurements. J. Geog. Res., 86, 457–463.

Yoon, S. W., Crum, L, A., Prosperetti, A., & Lu, N. Q. 1991. An investigation of the collective oscillations of a bubble cloud. *J. Acoust. Soc. of Amer.*, **89**(2), 700–6.

## REPORT DOCUMENTATION PAGE

Form Approved OPM No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and renaintaining the data needed, and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Information and Regulatory Affairs, Office of Management and Budget, Washington, DC 20503.

the Office of information and negulatory Allalis	, Office of Management and Budget, Washington					
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE January 1996	3. REPORT TYPE AND DA Final Technical Re				
4. TITLE AND SUBTITLE	January 1990	Tiliai Technicai Ke	5. FUNDING NUMBERS			
4. TITLE AND SUBTITLE  Low-Frequency Resonance Scattering from Acoustically Compact Bubble Clouds  ONR Grant N00014-93-1-07						
Low-Frequency Resonance S	ONK Grant N00014-95-1-0745					
6. AUTHOR(S)						
Ronald A. Roy and William I	M. Carey					
7. PERFORMING ORGANIZATION NAM	ME(S) AND ADDRESS(ES)		8. PERFORMING ORGANIZATION			
Applied Physics Laboratory			REPORT NUMBER			
University of Washington			· ·			
1013 NE 40th Street						
Seattle, WA 98105-6698						
9. SPONSORING / MONITORING AGEN	ICY NAME(S) AND ADDRESS(ES)		10. SPONSORING / MONITORING AGENCY REPORT NUMBER			
Office of Naval Research						
Ballston Centre Tower One						
800 North Quincy Street			**			
Arlington, VA 22217-5660						
11. SUPPLEMENTARY NOTES						
			_			
12a. DISTRIBUTION / AVAILABILITY S	TATEMENT		12b. DISTRIBUTION CODE			
Proposed: Approved for pub	lic release					
13. ABSTRACT (Maximum 200 words)						
	1 1 1 1 1 1 1 1	bbd com	to don'th. The question is: what role if			
Microbubble plumes, layers and clouds are produced when waves break and are convected to depth. The question is: what role, if any, is played by these microbubble plumes in the scattering of sound near the sea surface in the low (20 Hz) to mid (2 kHz) fre-						
any, is played by these micro	bubble plumes in the scattering of	r sound near the sea surface	in the low (20 Hz) to find (2 kHz) he-			
			esonant oscillators, then radiated noise. This hypothesis is based on the theory			
			cy of excitation are present in an acous-			
ticelly compact region, the mi	vivre properties determine the radi	iotion and scattering. We pre	sent scattering measurements from sub-			
merged bubble clouds exhibi	ting low-frequency resonances w	vith the lowest order mode b	peing a monopole corresponding to the			
merged bubble clouds exhibiting low-frequency resonances, with the lowest order mode being a monopole corresponding to the volumetric pulsation of the cloud. Moreover, when such a cloud is located near the sea surface, a dipole radiation pattern results.						
Consequently, the scattered intensity is a sensitive function of depth, frequency, surface roughness and grazing angle. We show						
that for certain sea state conditions, observable anomalous effects can result from transient plumes generated close to the surface.						
that for cortain son state constitutes, coor raise anomalous errors can result with a tansient prairies generates erest to the states.						
14. SUBJECT TERMS			15. NUMBER OF PAGES			
Bubbles, Bubble Clouds, Res						
Scattering Experiment, Parar						
Scattering Experiment, Furthern Scattering	,	,				
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	ON 20. LIMITATION OF ABSTRACT			
Unclassified	Unclassified	Unclassified				